Minería de datos y Patrones

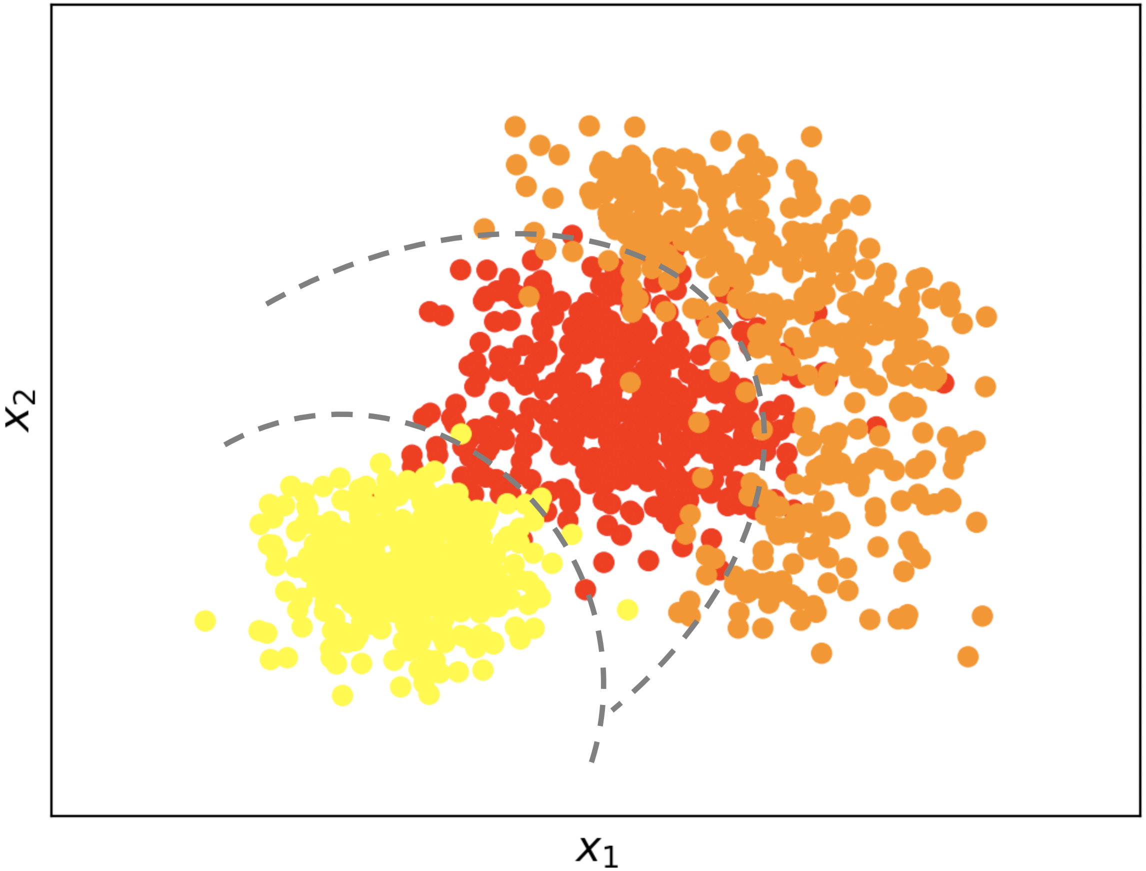
Version 2024-1

Neural Networks

[Capítulo 4]

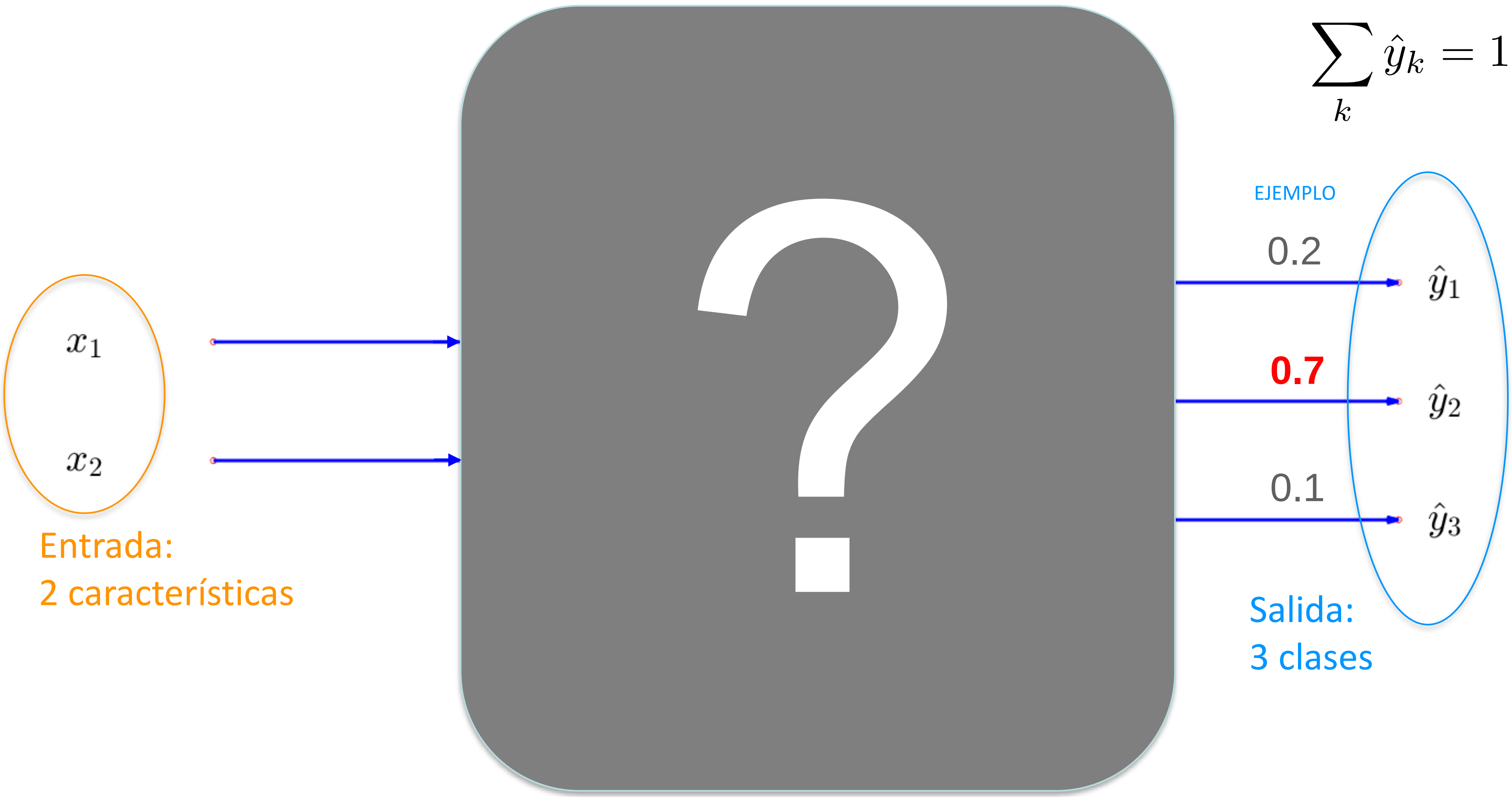
Dr. José Ramón Iglesias
DSP-ASIC BUILDER GROUP
Director Semillero TRIAC
Ingenieria Electronica
Universidad Popular del Cesar

Ejemplo: ¿cómo sería una red neuronal de 2 características y 3 clases?



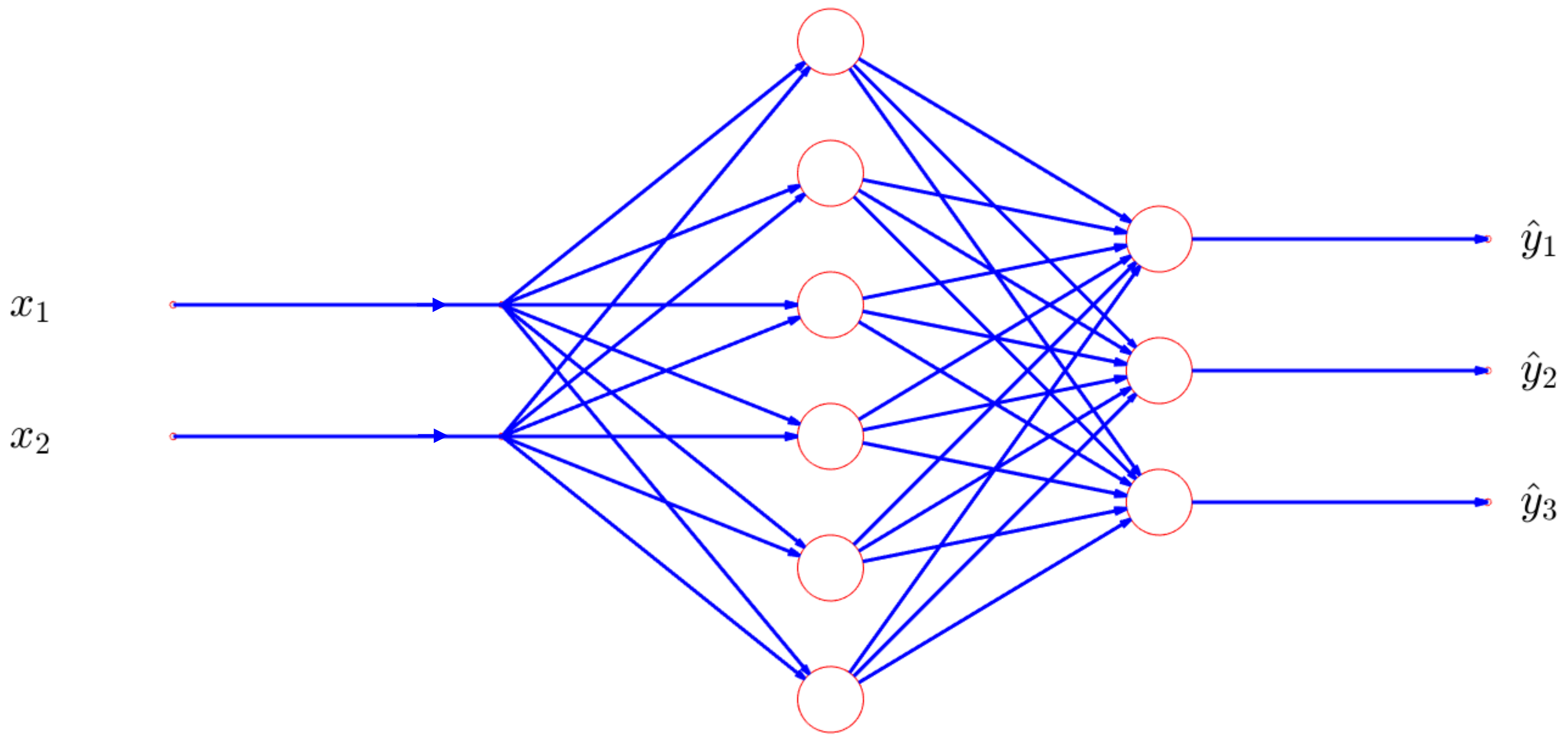
Ejemplo: ¿cómo sería una red neuronal de 2 características y 3 clases?

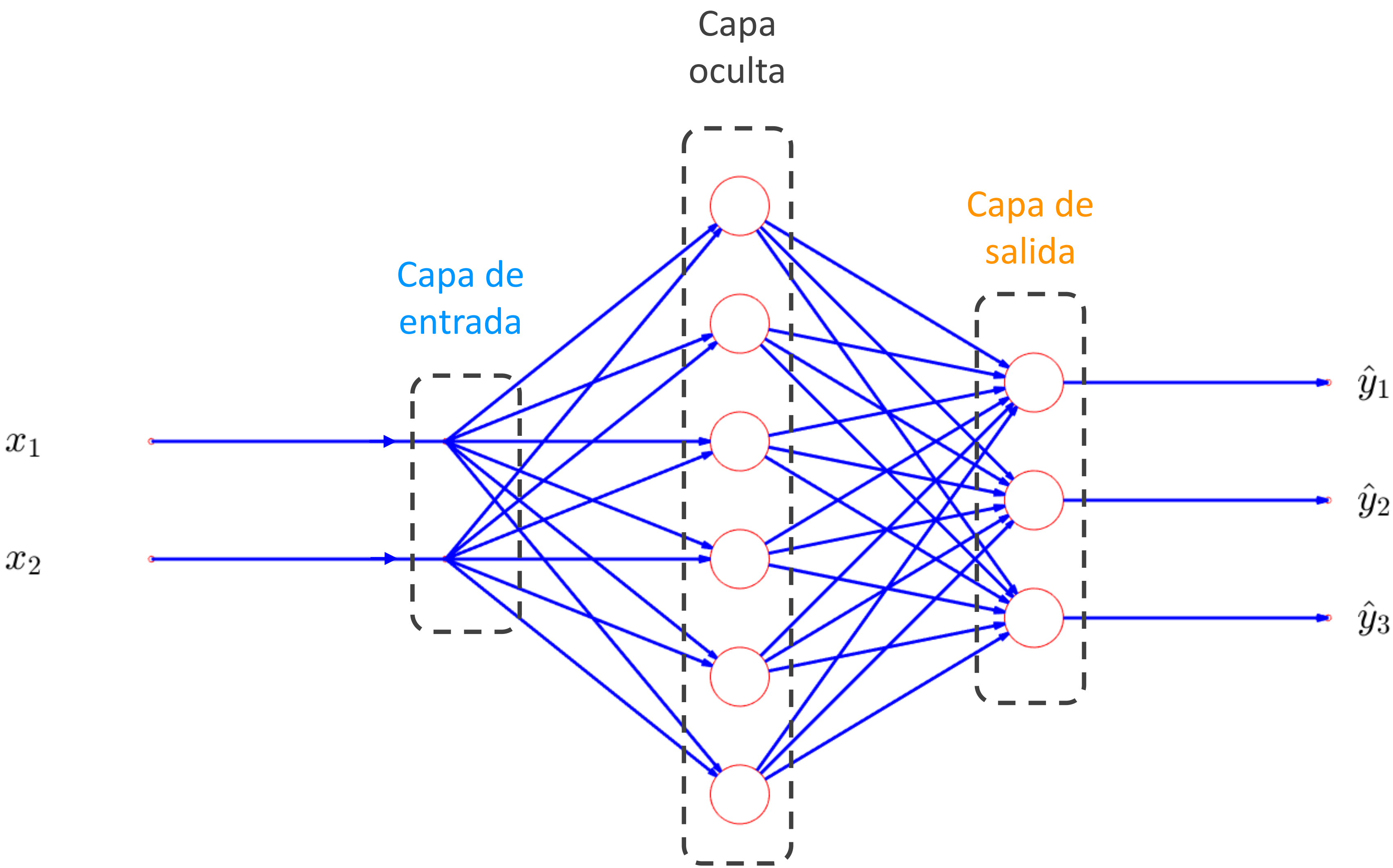
$$\sum_k \hat{y}_k = 1$$

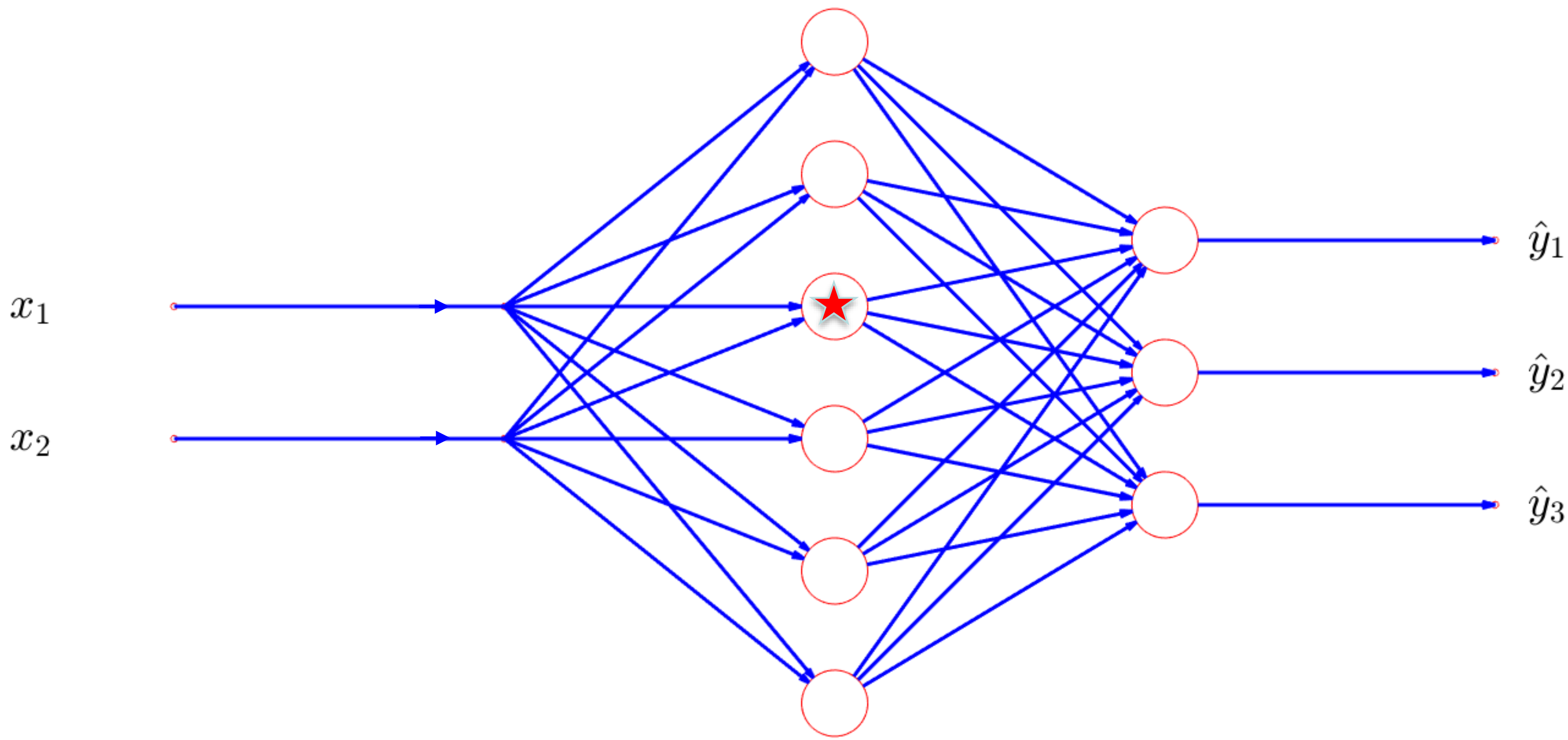


Clasificación: $\hat{k} = argmax_k[\hat{y}_k]$ = 2

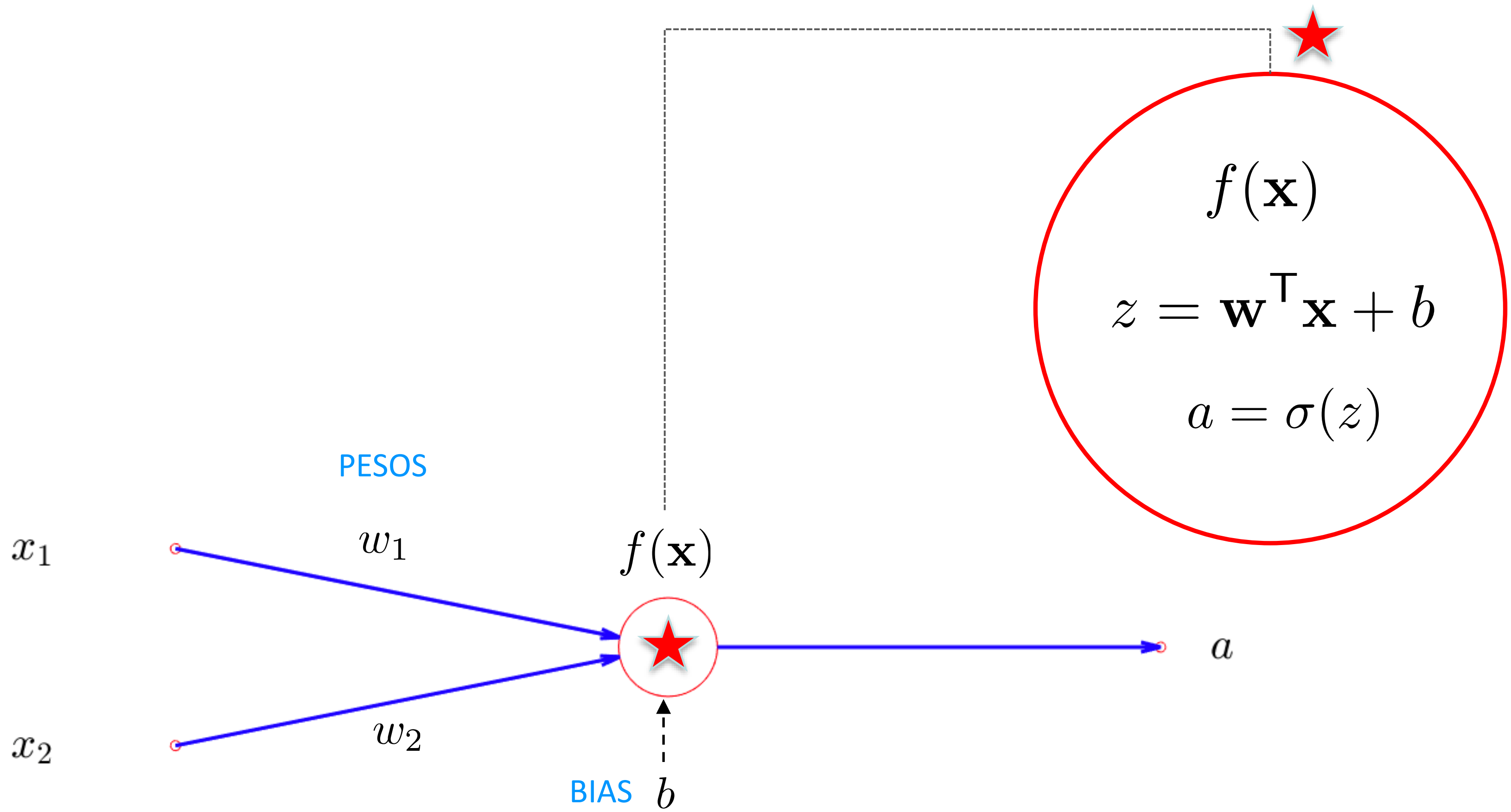
Ejemplo: ¿cómo sería una red neuronal de 2 características y 3 clases?




Capa
oculta
Capa de
salida
Capa de
entrada
x_1
x_2
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$



x_1
x_2
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$



$f(\mathbf{x}) = a$

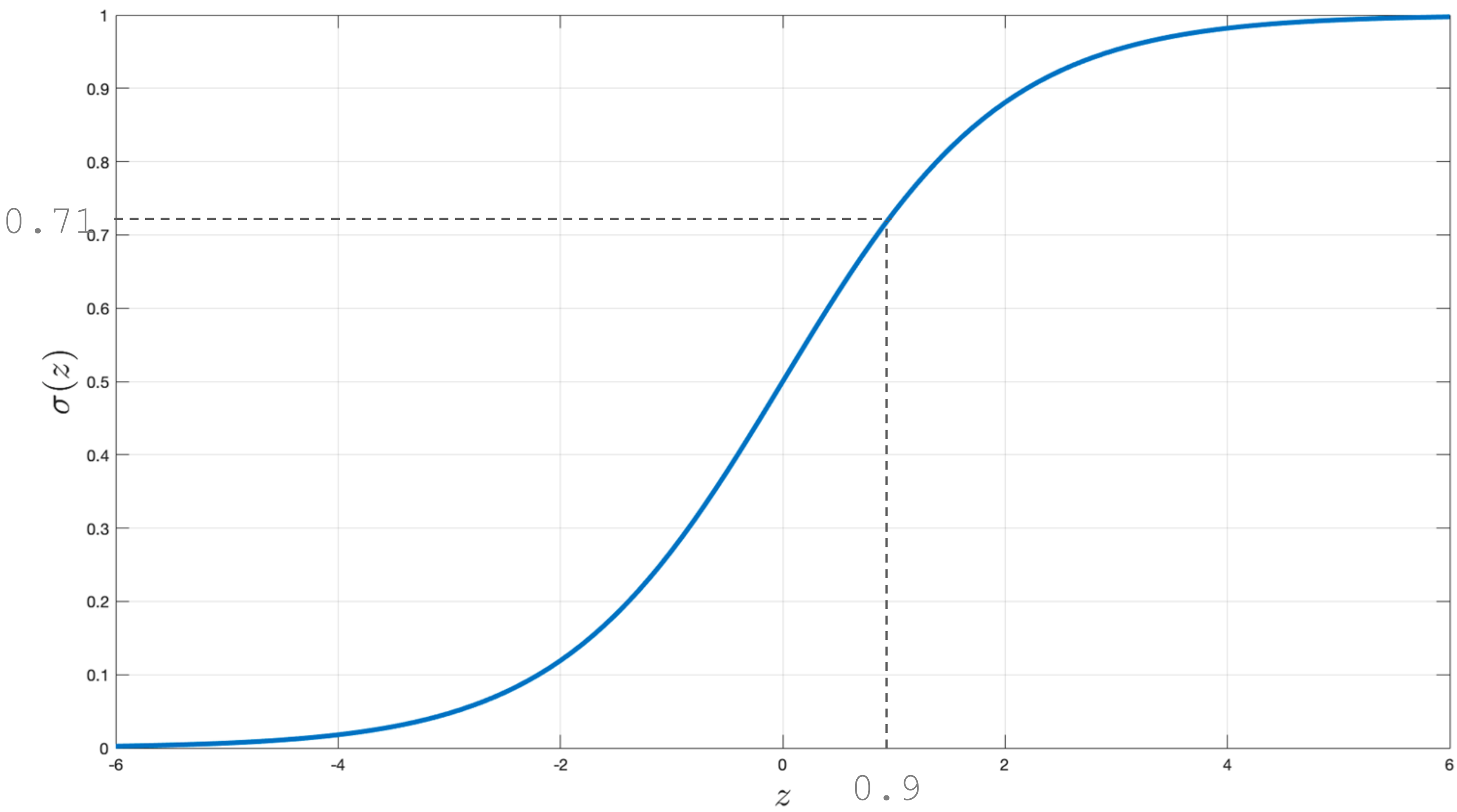
1) $z = w_1 x_1 + w_2 x_2 + b \ = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b$

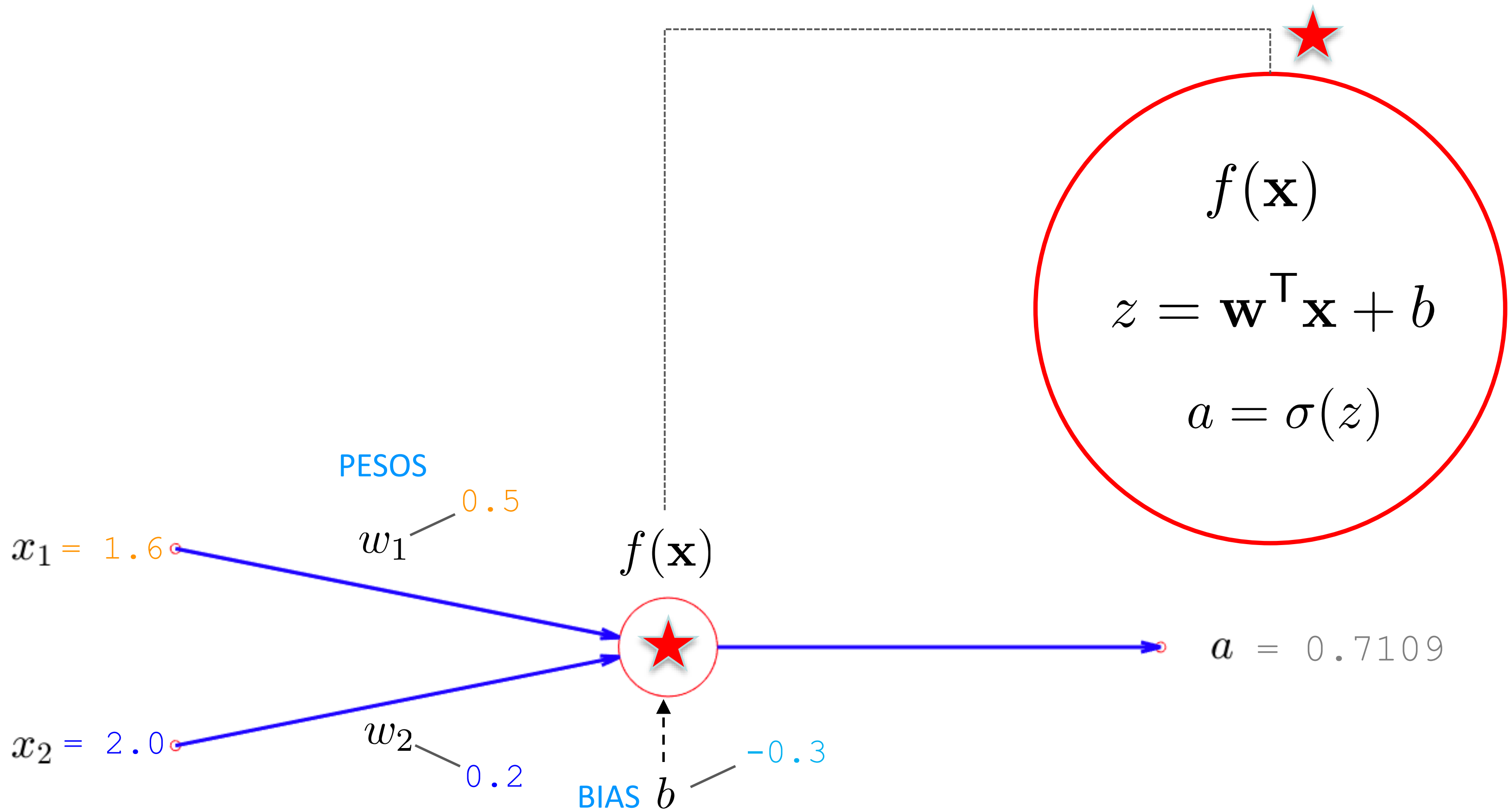
2) $a = \sigma(z)$

FUNCIÓN
NO-LIENAL

SIGMOIDE

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



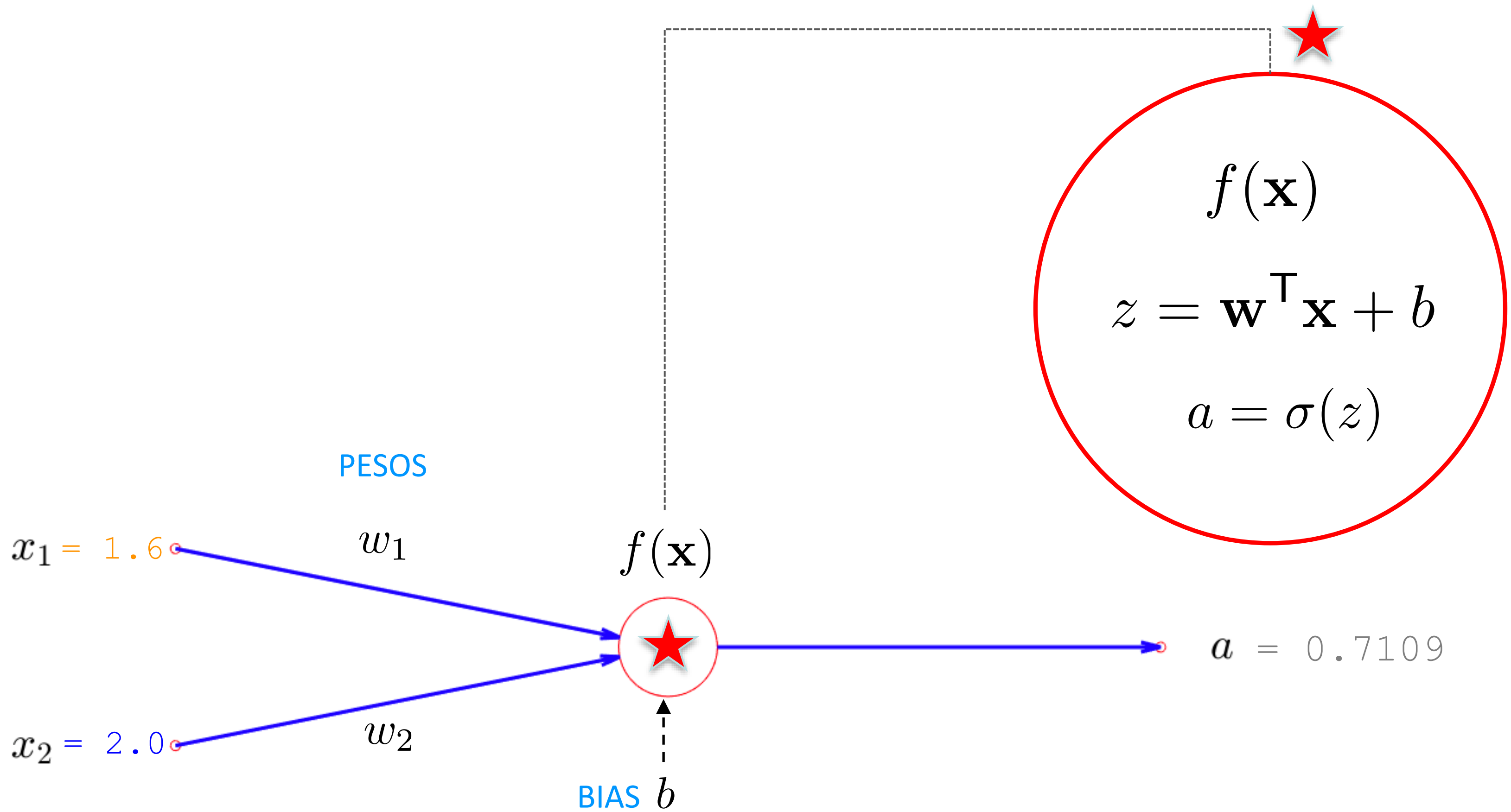


$$f(\mathbf{x}) = a$$

1) $z = \overset{0.8}{w_1 x_1} + \overset{0.4}{w_2 x_2} + \overset{-0.3}{b} = \mathbf{w}^\mathsf{T}\mathbf{x} + b = 0.9$

2) $a = \sigma(z) = 0.7109$

FUNCIÓN
NO-LIENAL



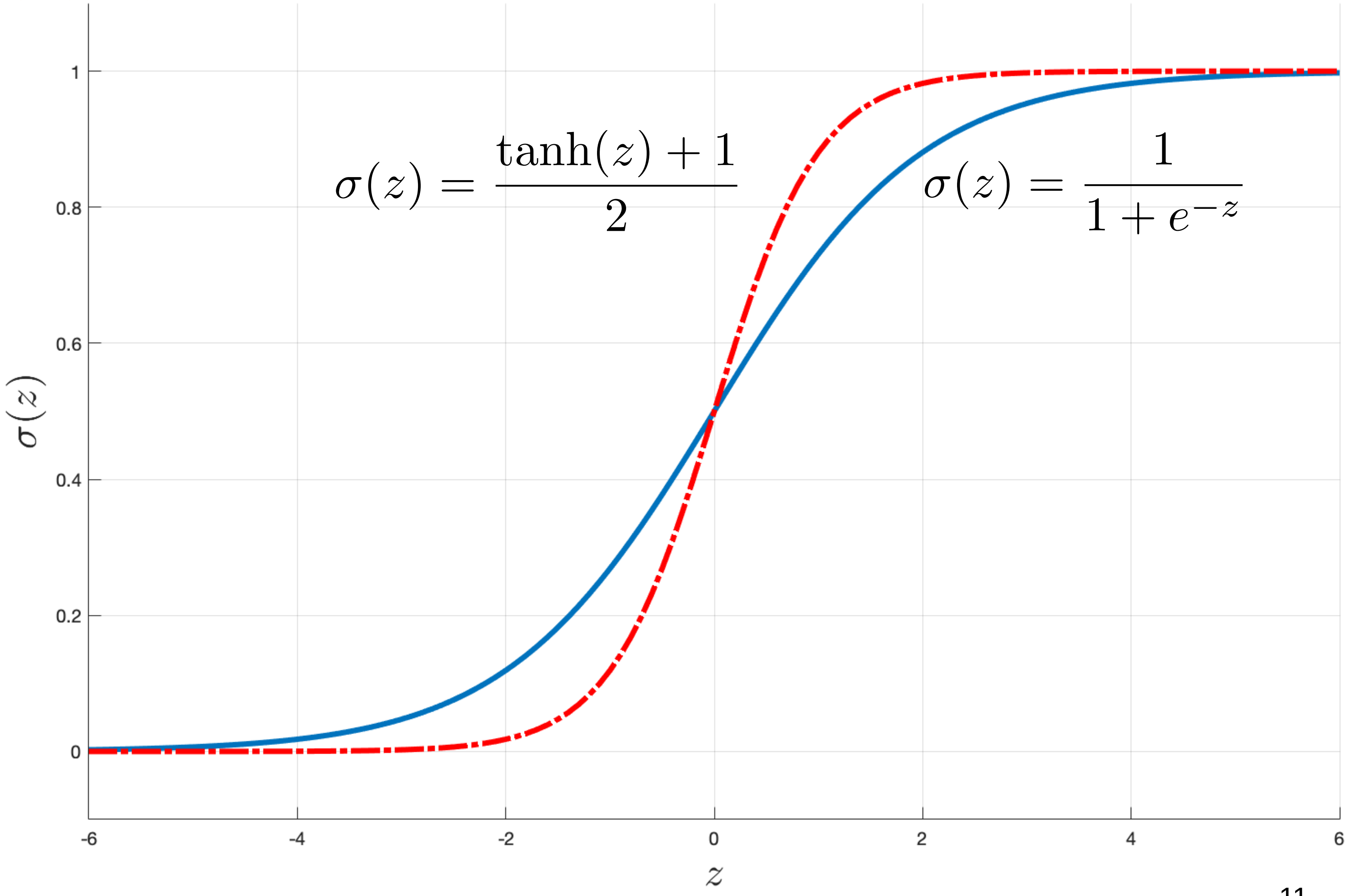
$$f(\mathbf{x}) = a$$

1) $z = w_1 x_1 + w_2 x_2 + b \ = \mathbf{w}^\mathsf{T}\mathbf{x} + b$

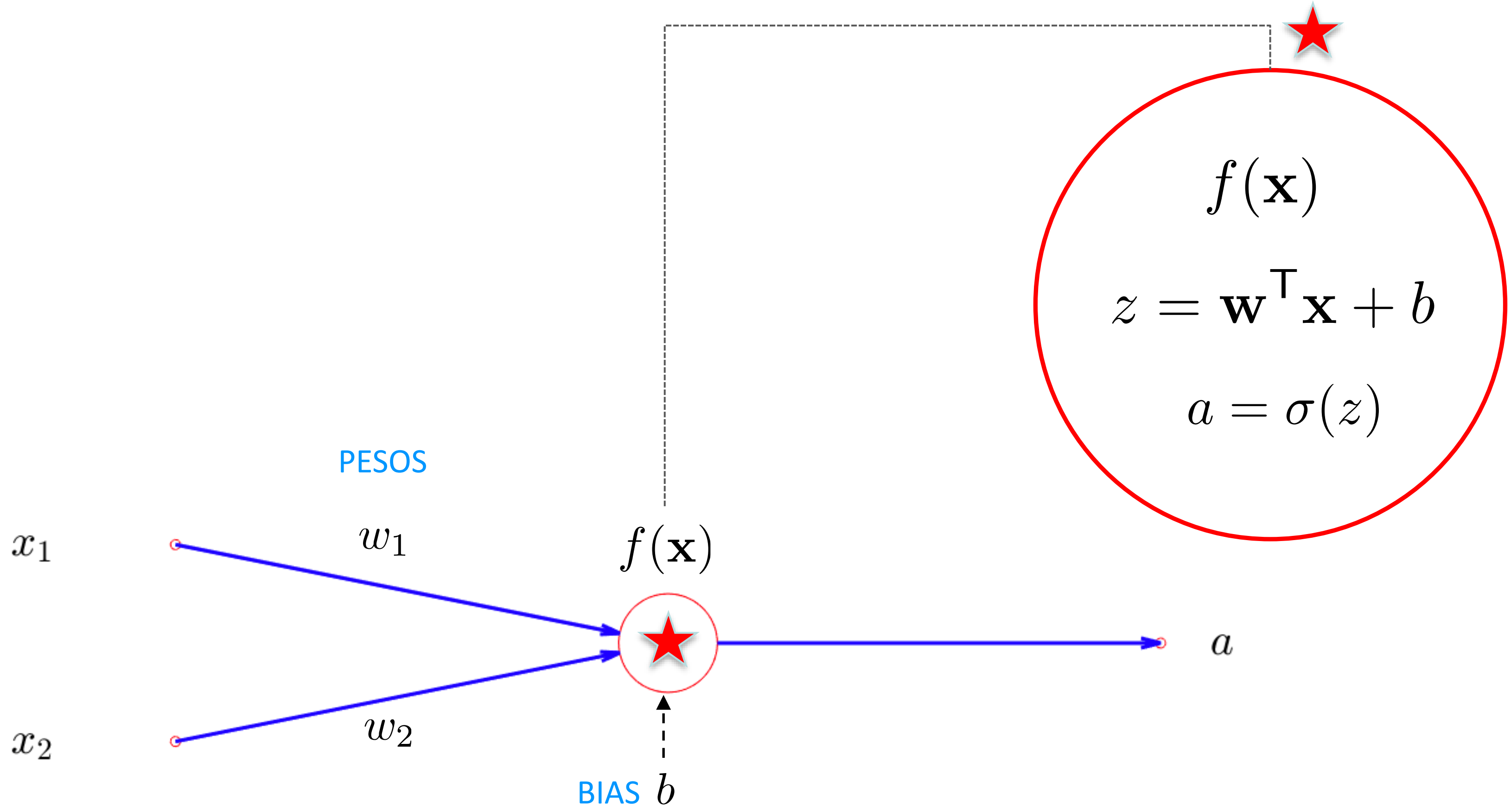
2) $a = \sigma(z)$

FUNCIÓN
NO-LIENAL

SIGMOIDE

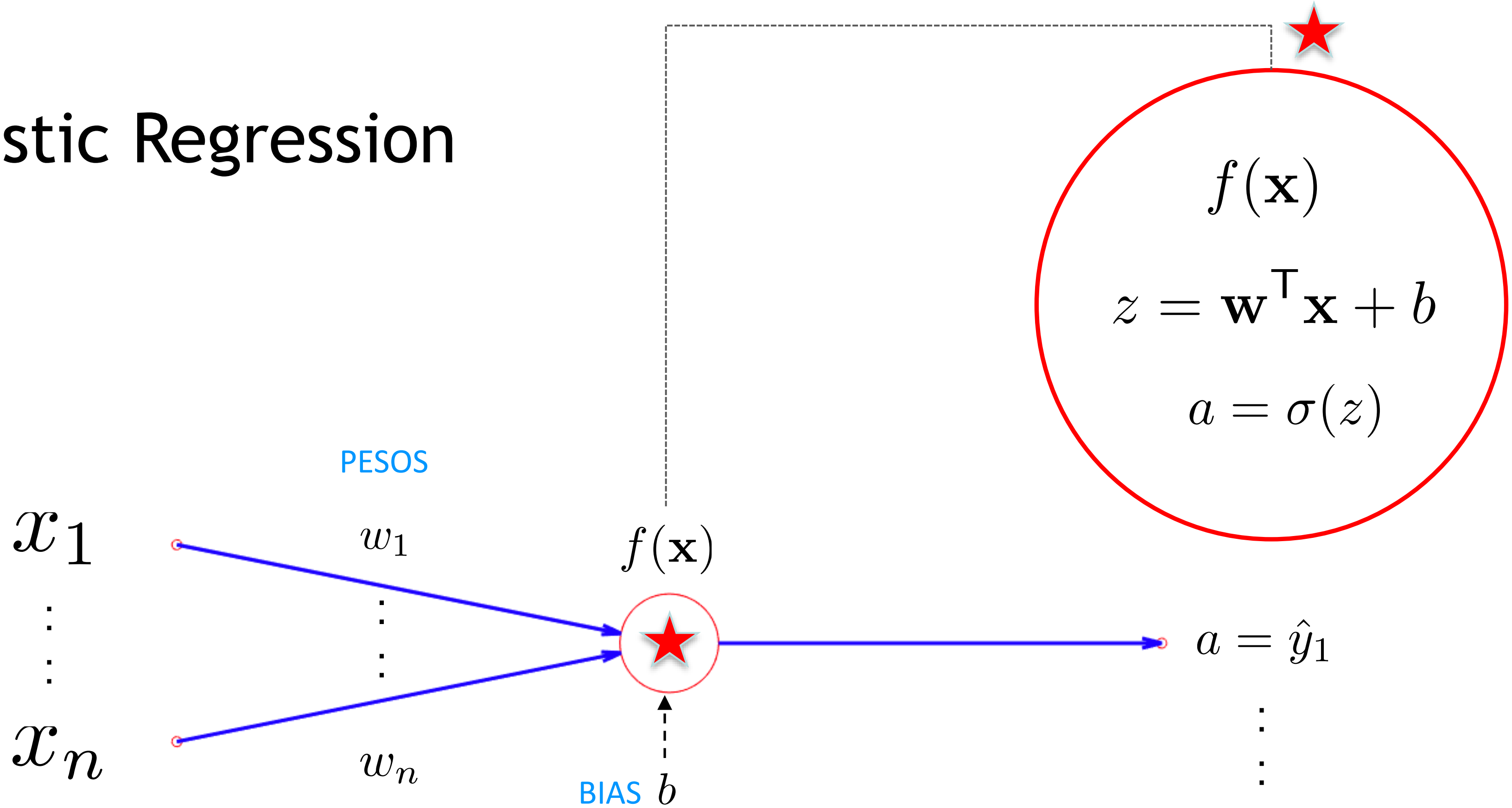


$\sigma(z) = \frac{\tanh(z)+1}{2}$
$\sigma(z) = \frac{1}{1+e^{-z}}$
1
0.8
0.6
0.4
0.2
0
-6
-4
-2
0
2
4
6
$\sigma(z)$
z



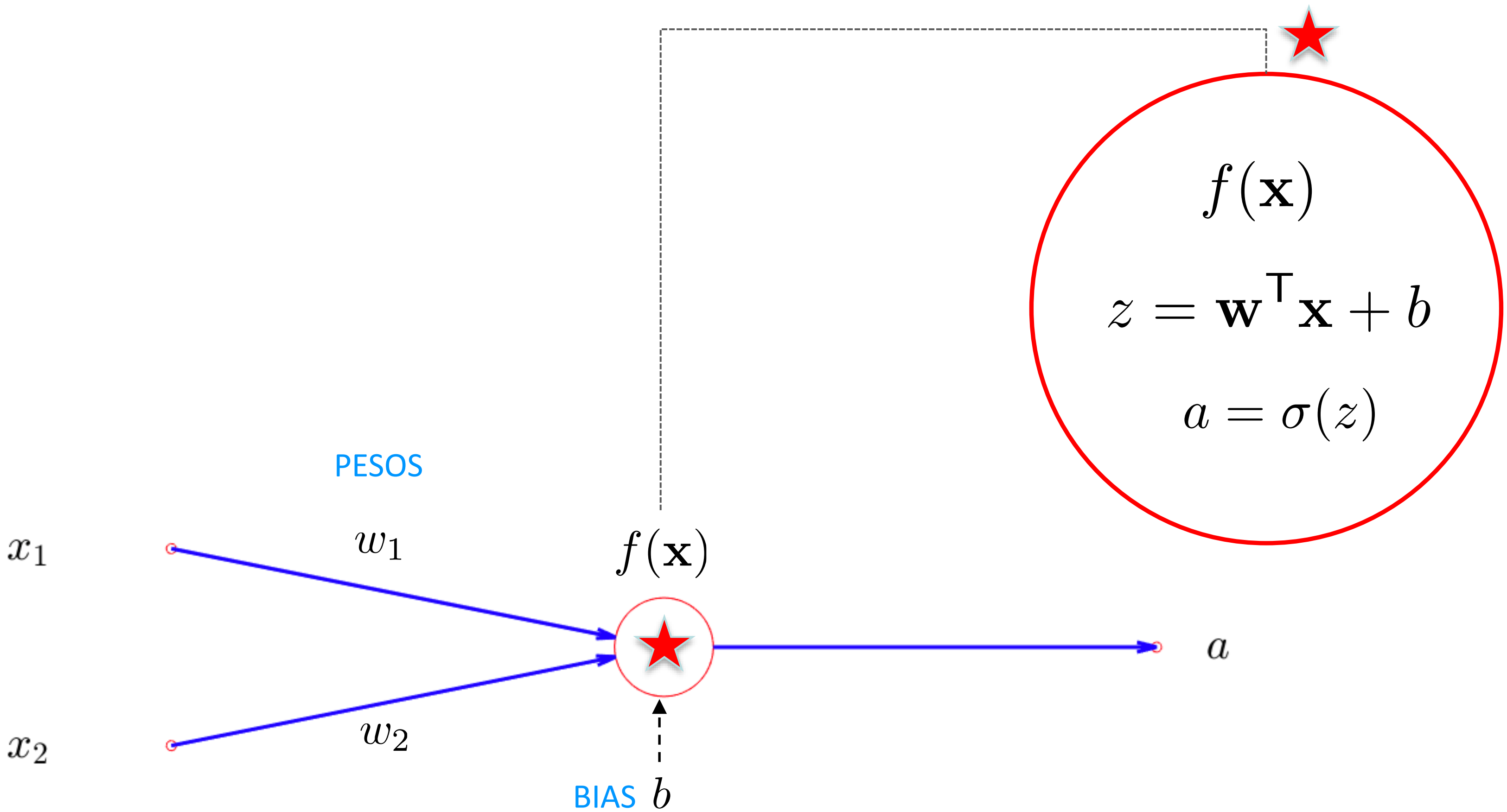
$f(\mathbf{x})$
$z = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b$
$a = \sigma(z)$
PESOS
x_1
w_1
$f(\mathbf{x})$
x_2
w_2
a
BIAS b

Logistic Regression

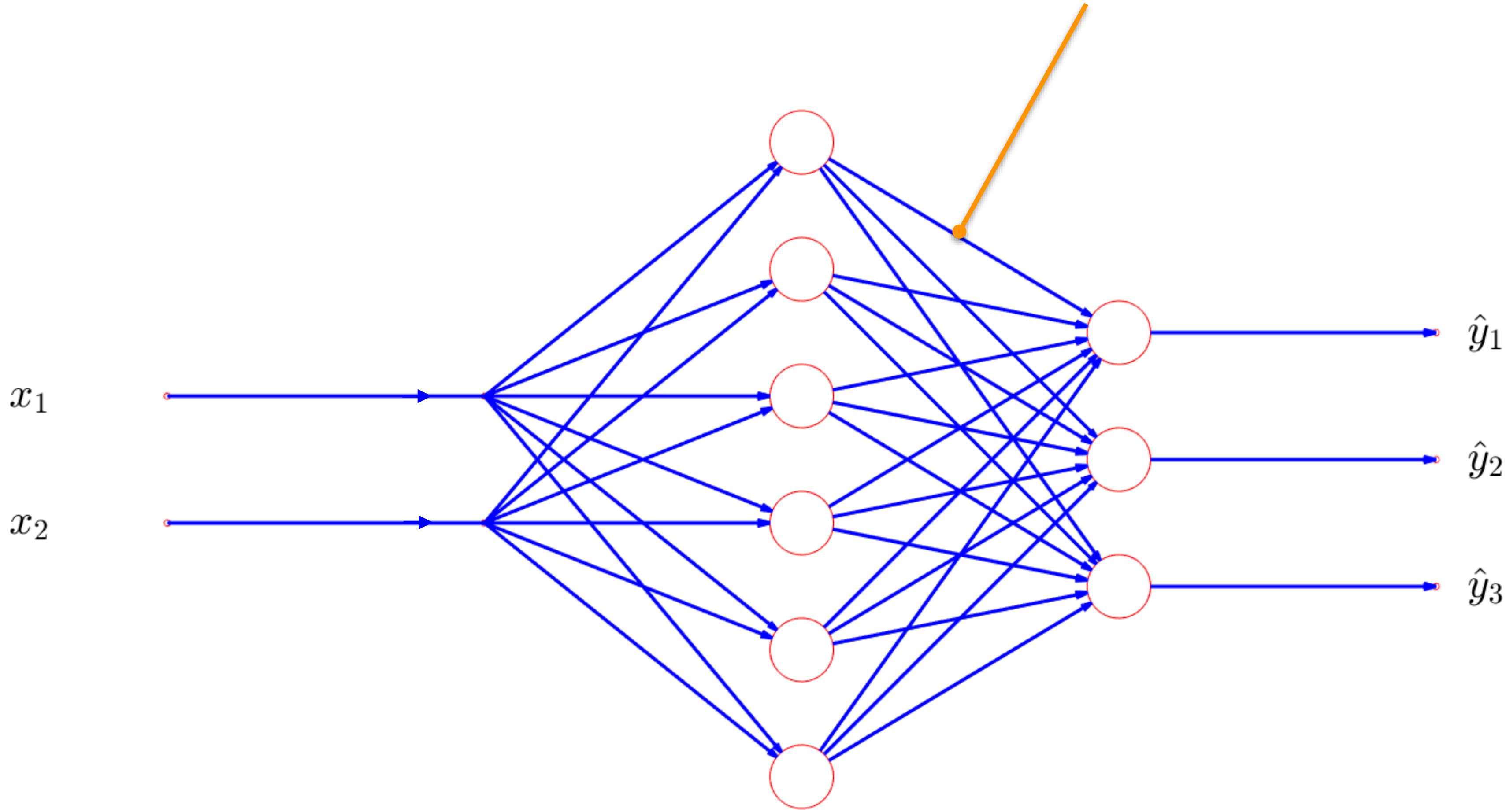


Los parámetros se encuentran minimizando una función de pérdida, por ejemplo:

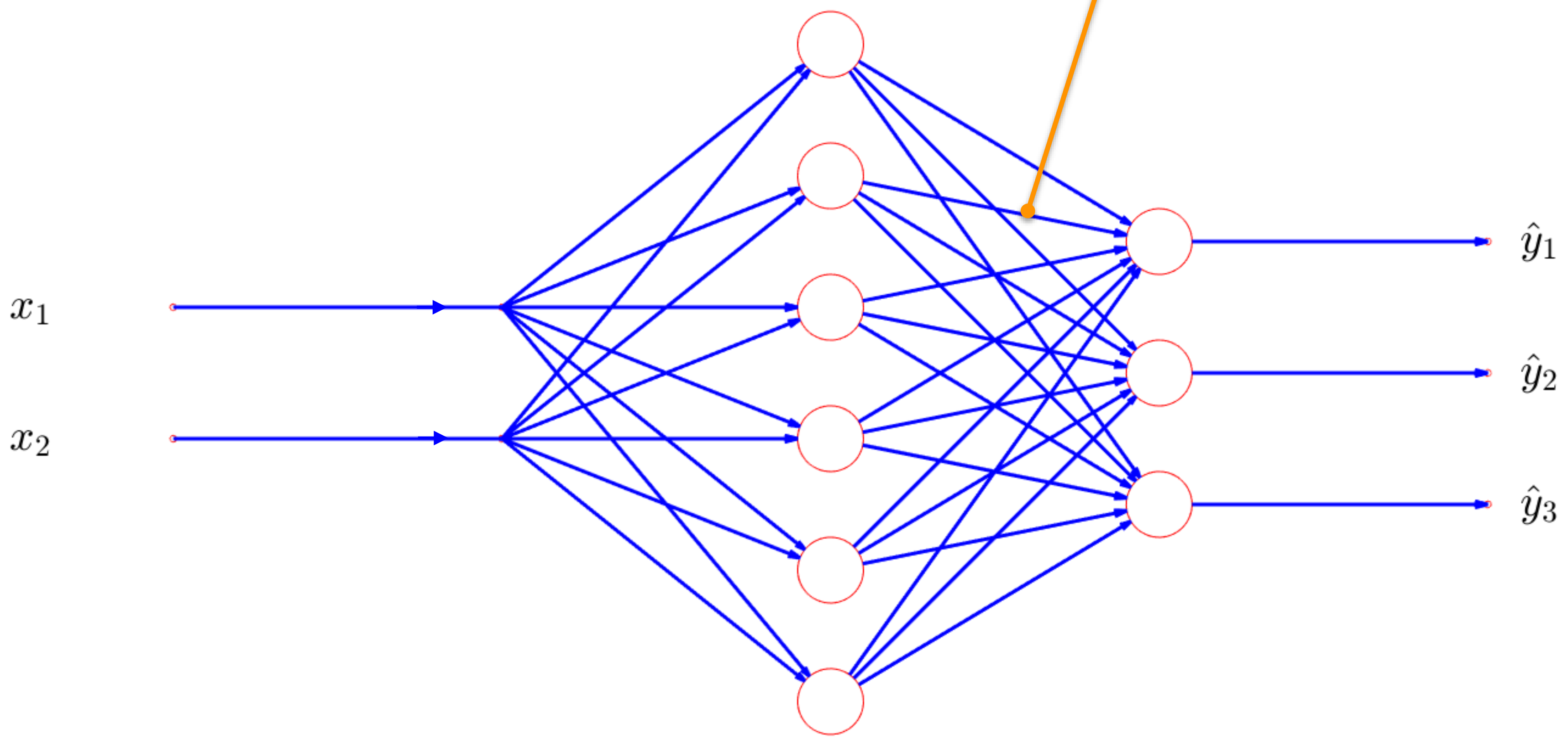
$$J(w_1, \cdots, w_n, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$


$f(\mathbf{x})$
$z = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b$
$a = \sigma(z)$
PESOS
x_1
w_1
$f(\mathbf{x})$
a
x_2
w_2
BIAS b

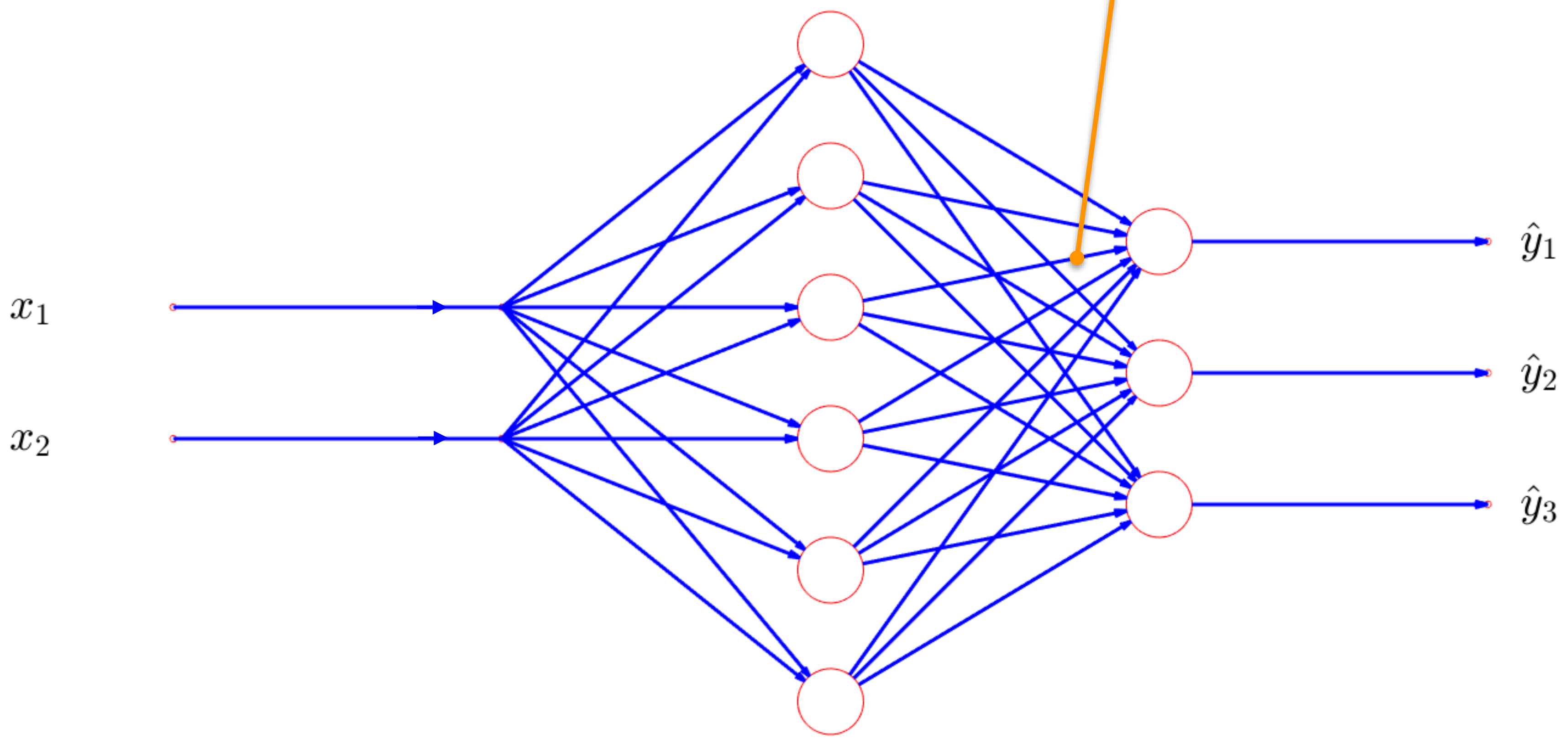
$$a_{11} = \sigma(w_{11}x_1 + w_{21}x_2 + b_{11})$$



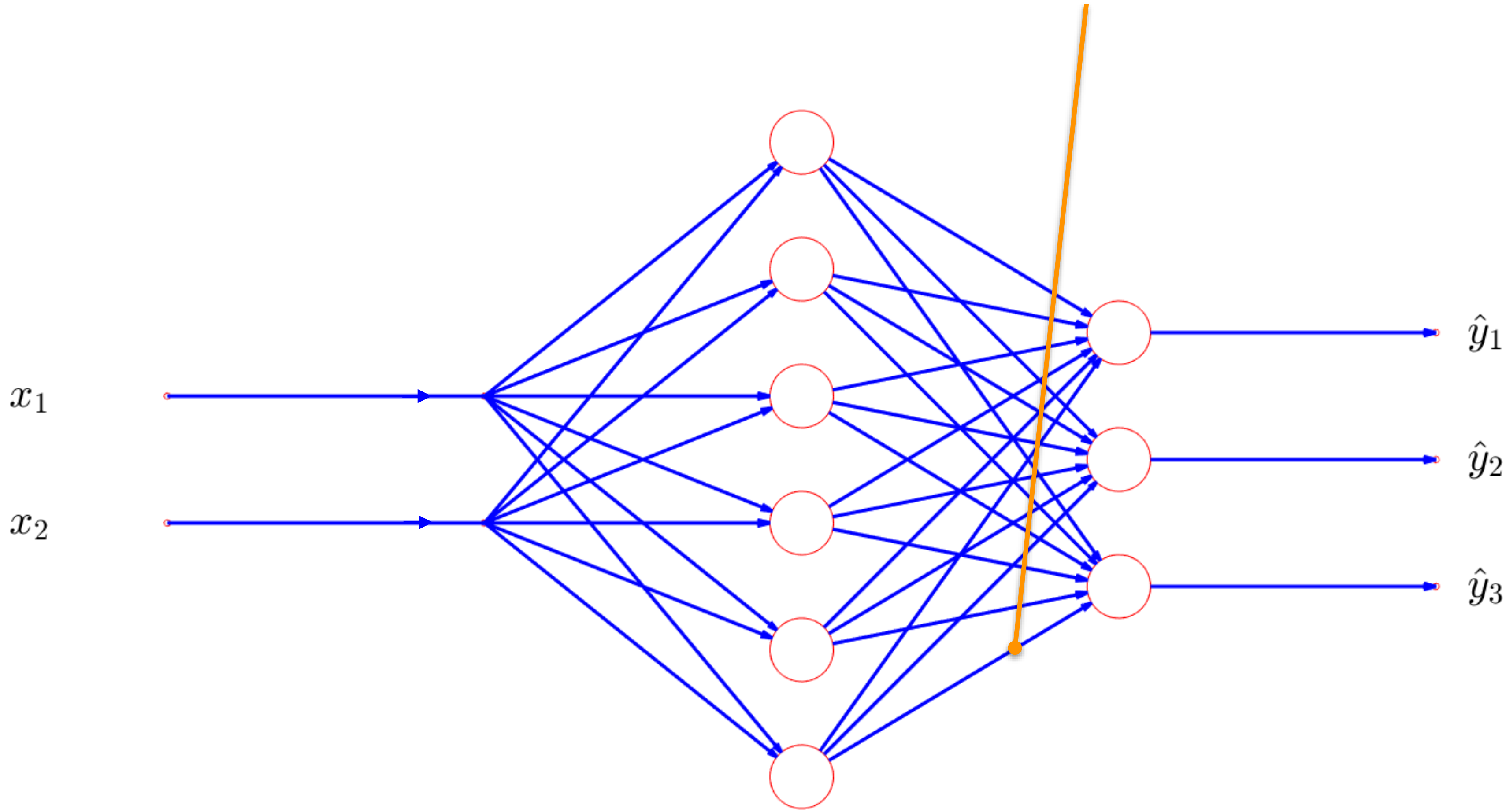
$$a_{12} = \sigma(w_{12}x_1 + w_{22}x_2 + b_{12})$$



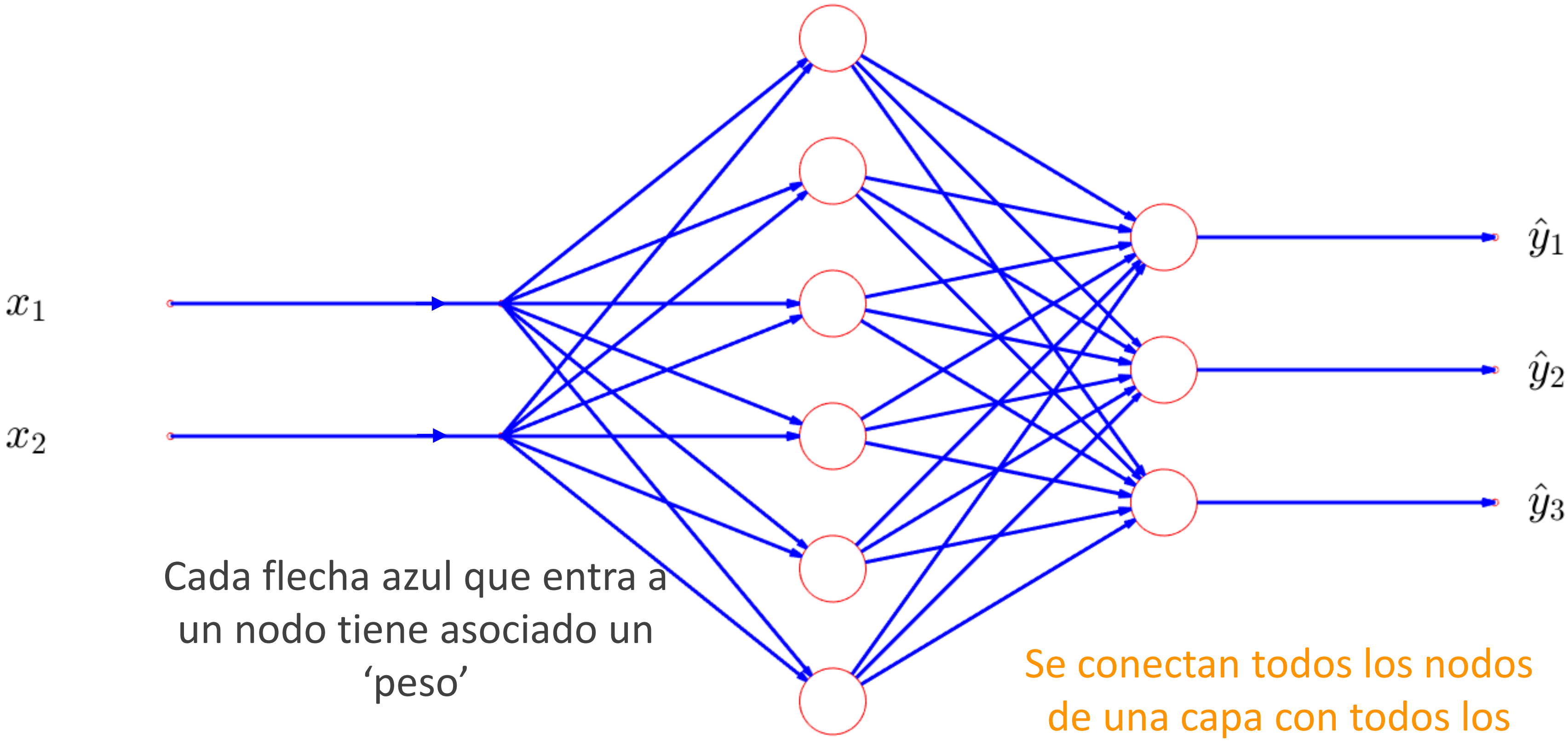
$$a_{13} = \sigma(w_{13}x_1 + w_{23}x_2 + b_{13})$$



$$a_{16} = \sigma(w_{16}x_1 + w_{26}x_2 + b_{16})$$



Cada nodo (círculo rojo)
tiene asociado una función
no-lineal y un valor de ‘bias’

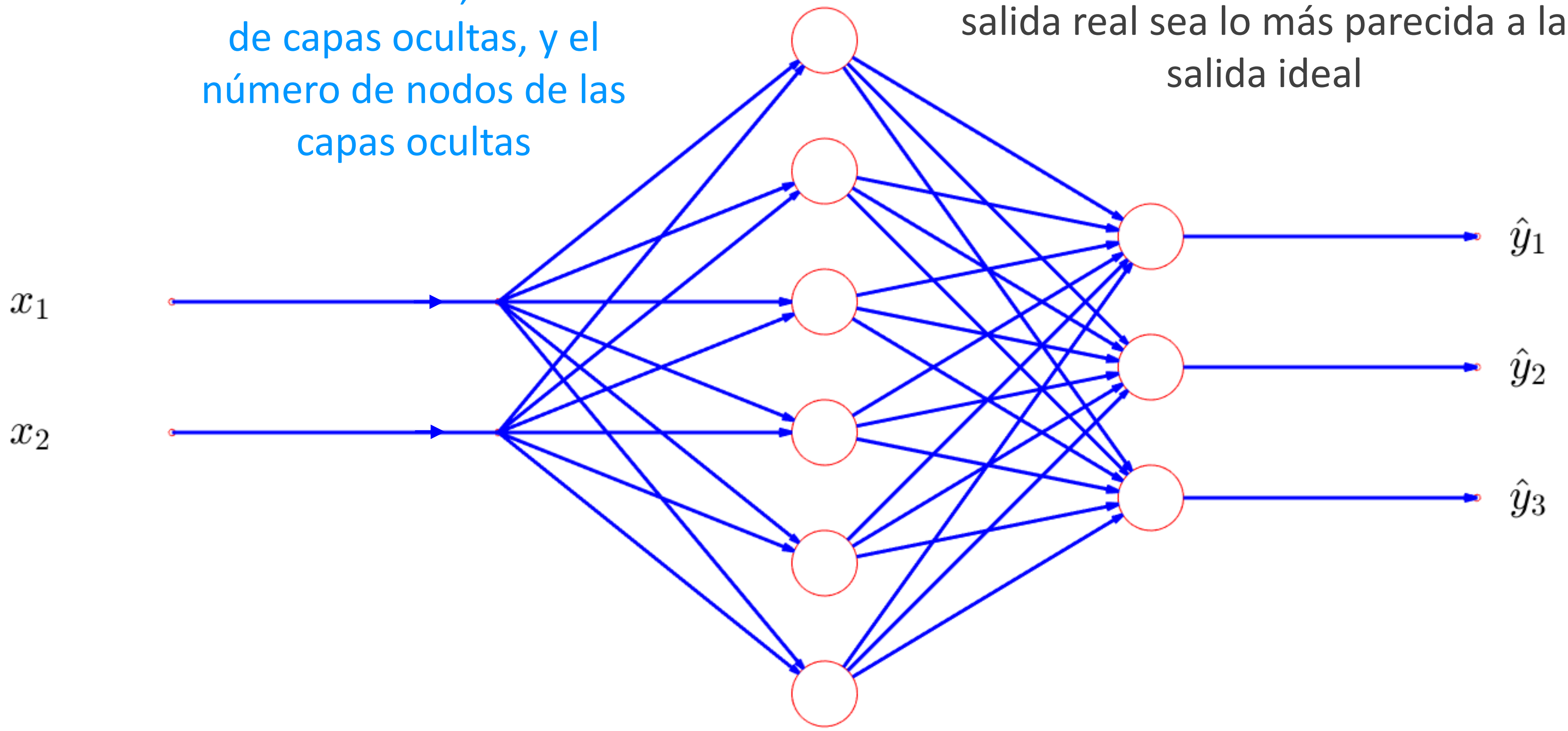


Cada flecha azul que entra a
un nodo tiene asociado un
‘peso’

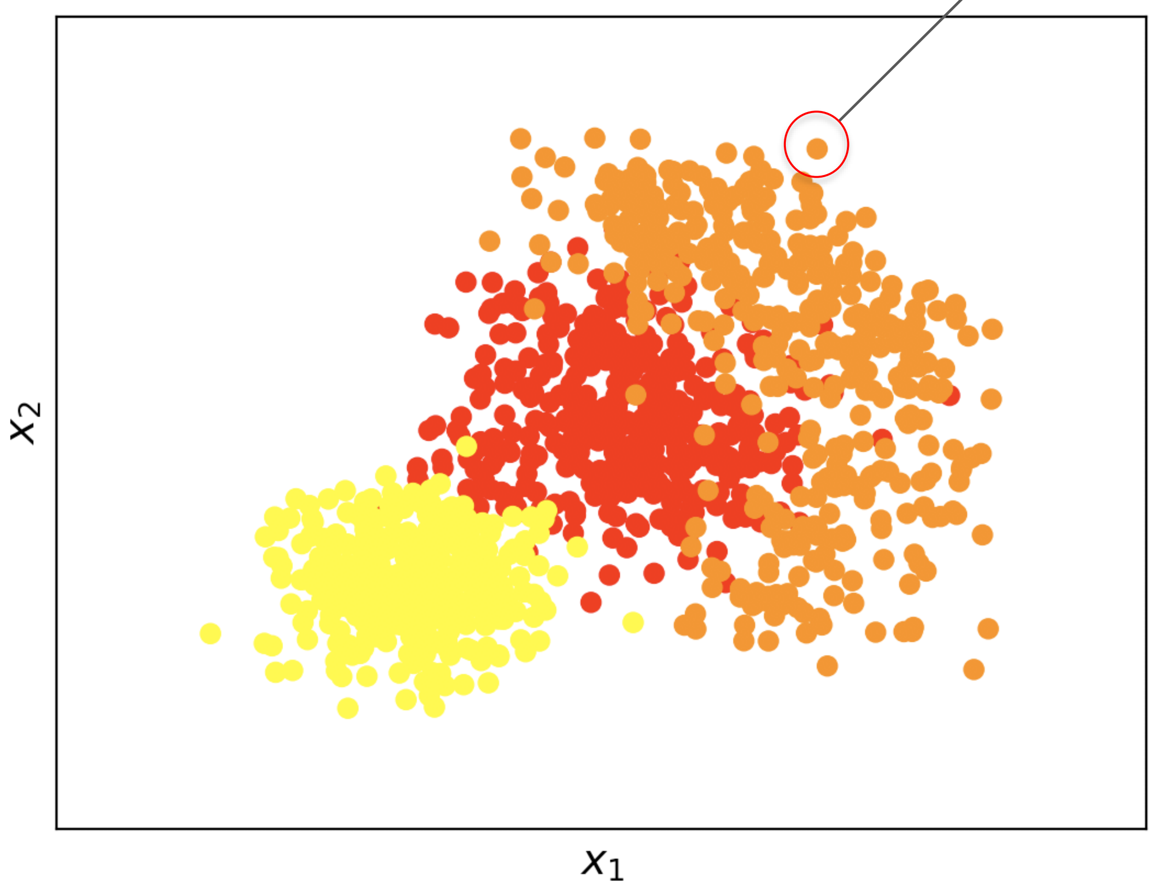
Se conectan todos los nodos
de una capa con todos los
nodos de la siguiente capa

Cuando se diseña una red neuronal se debe decidir la función no lineal, el número de capas ocultas, y el número de nodos de las capas ocultas

Cuando se entrena una red neuronal se debe estimar todos los pesos y los 'bias' de tal forma que la salida real sea lo más parecida a la salida ideal



$$\mathbf{x} = (1.6, 2.0)$$



como pertenece a clase '3', entonces su salida ideal es:

$$\mathbf{y} = (0, 0, 1)$$

es posible que la salida real de la red sea:

$$\hat{\mathbf{y}} = (0.1, 0.2, 0.7)$$

como el máximo de la salida es el tercer elemento, entonces la clasificación será **Clase '3'**

En el entrenamiento se debe minimizar una función objetivo del tipo:

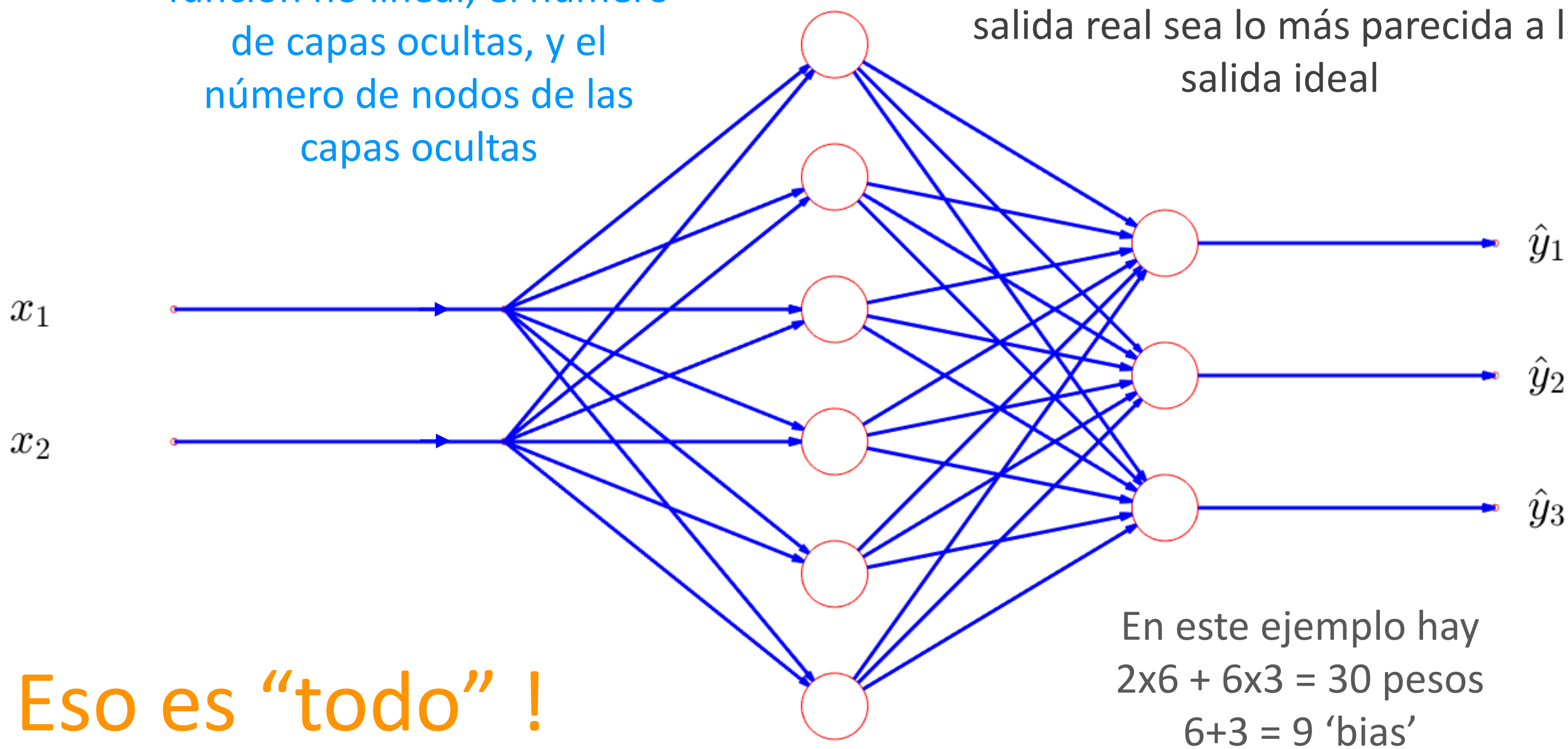
$$J(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||\hat{\mathbf{y}}_i - \mathbf{y}_i||^2$$

Θ Parámetros (pesos y 'bias' de todas las capas)

N Número de muestras del set de entrenamiento

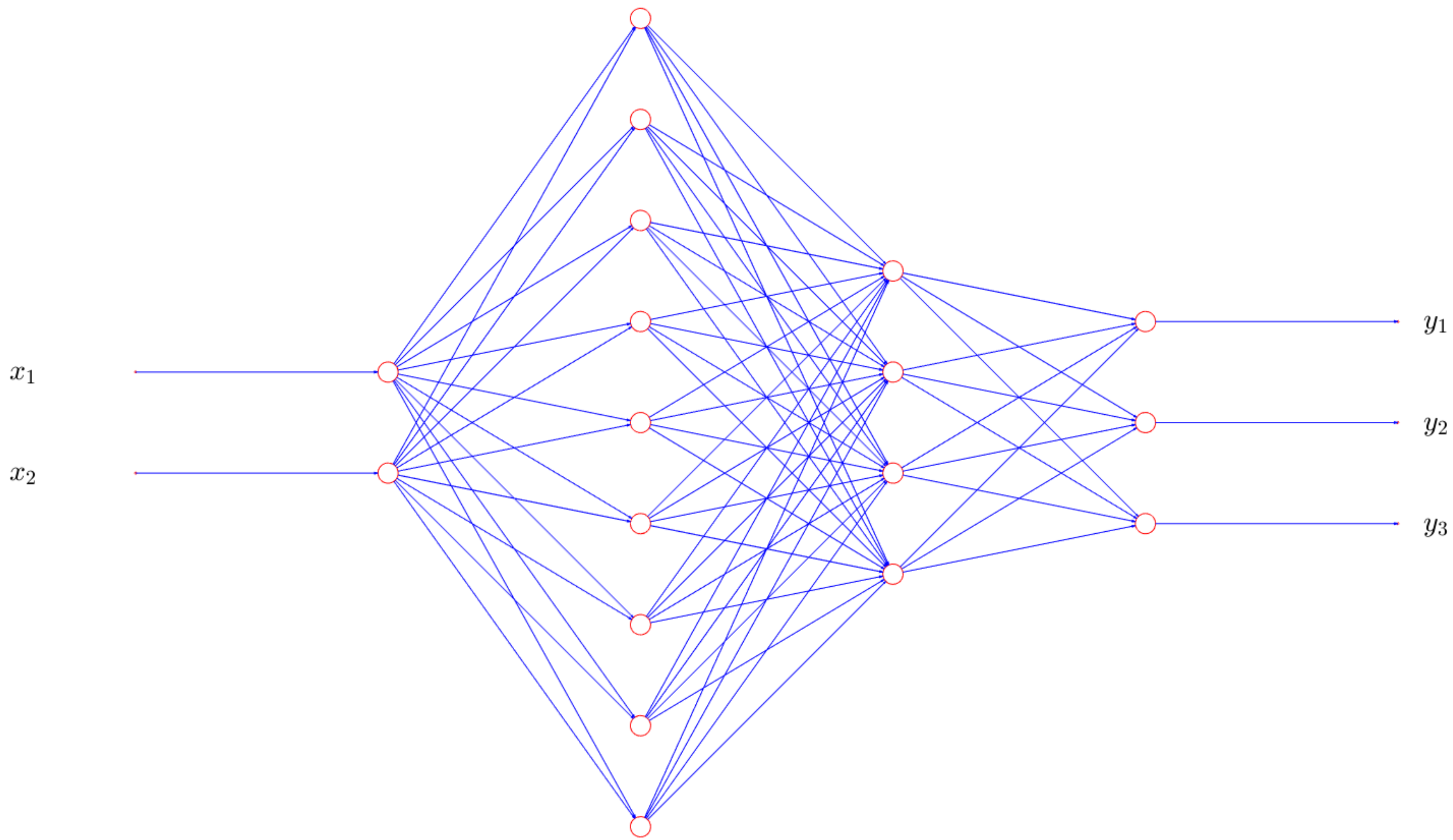
Cuando se diseña una red neuronal se debe decidir la función no lineal, el número de capas ocultas, y el número de nodos de las capas ocultas

Cuando se entrena una red neuronal se debe estimar todos los pesos y los ‘bias’ de tal forma que la salida real sea lo más parecida a la salida ideal

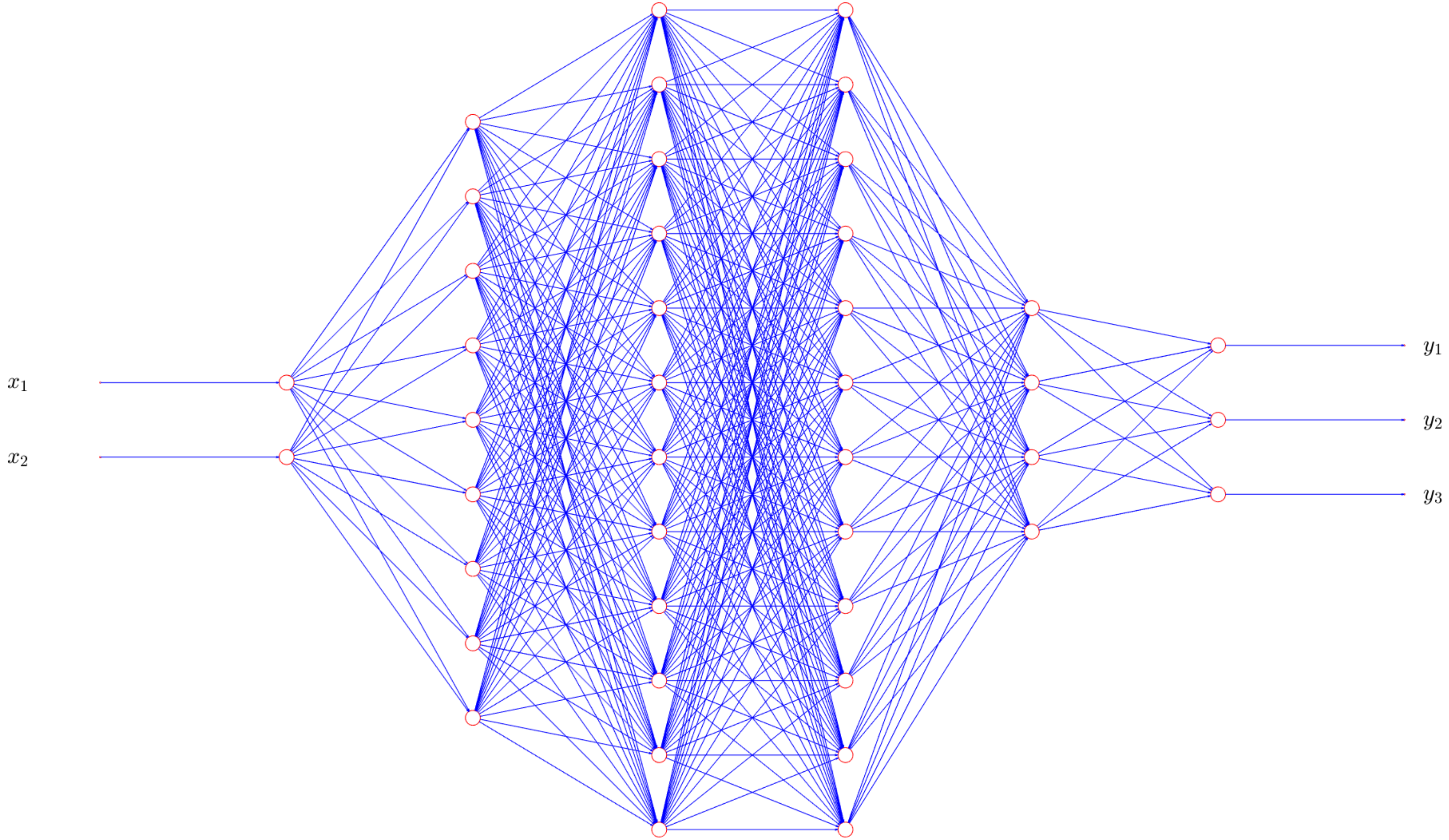


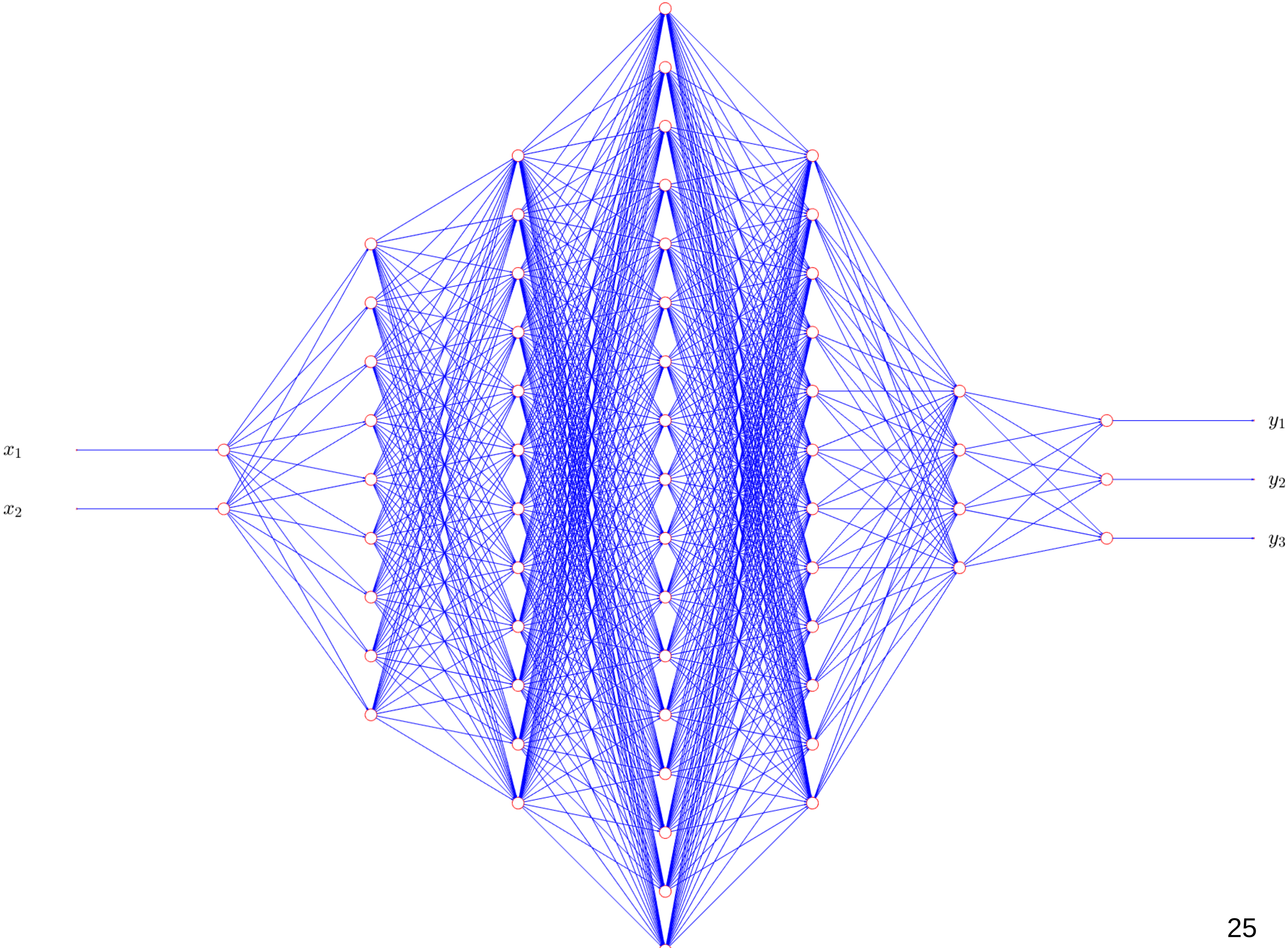
Eso es “todo” !

En este ejemplo hay
2x6 + 6x3 = 30 pesos
6+3 = 9 ‘bias’
Es decir, 39 parámetros a estimar en el entrenamiento

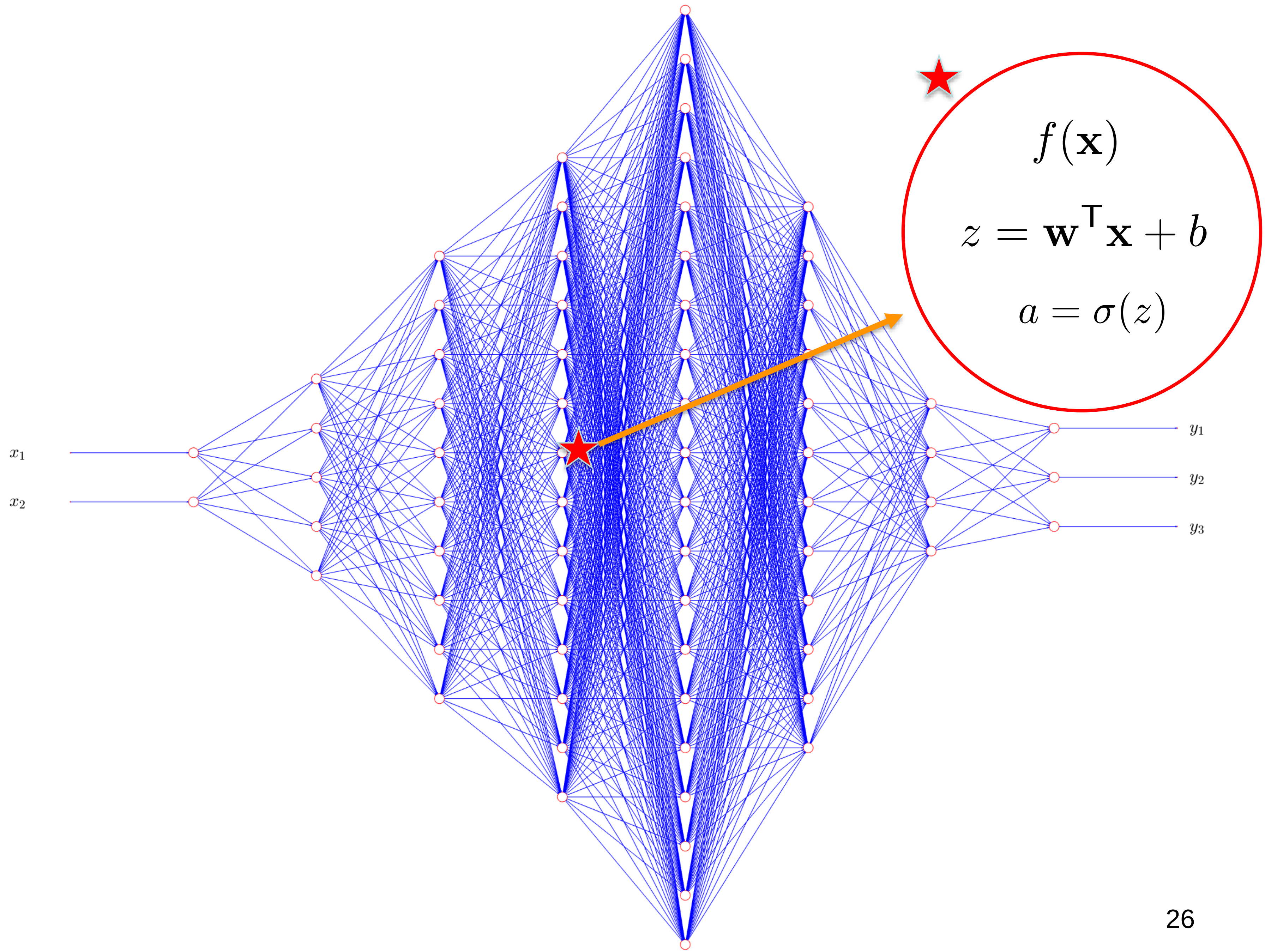


x_1
x_2
y_1
y_2
y_3


x_1
x_2
y_1
y_2
y_3



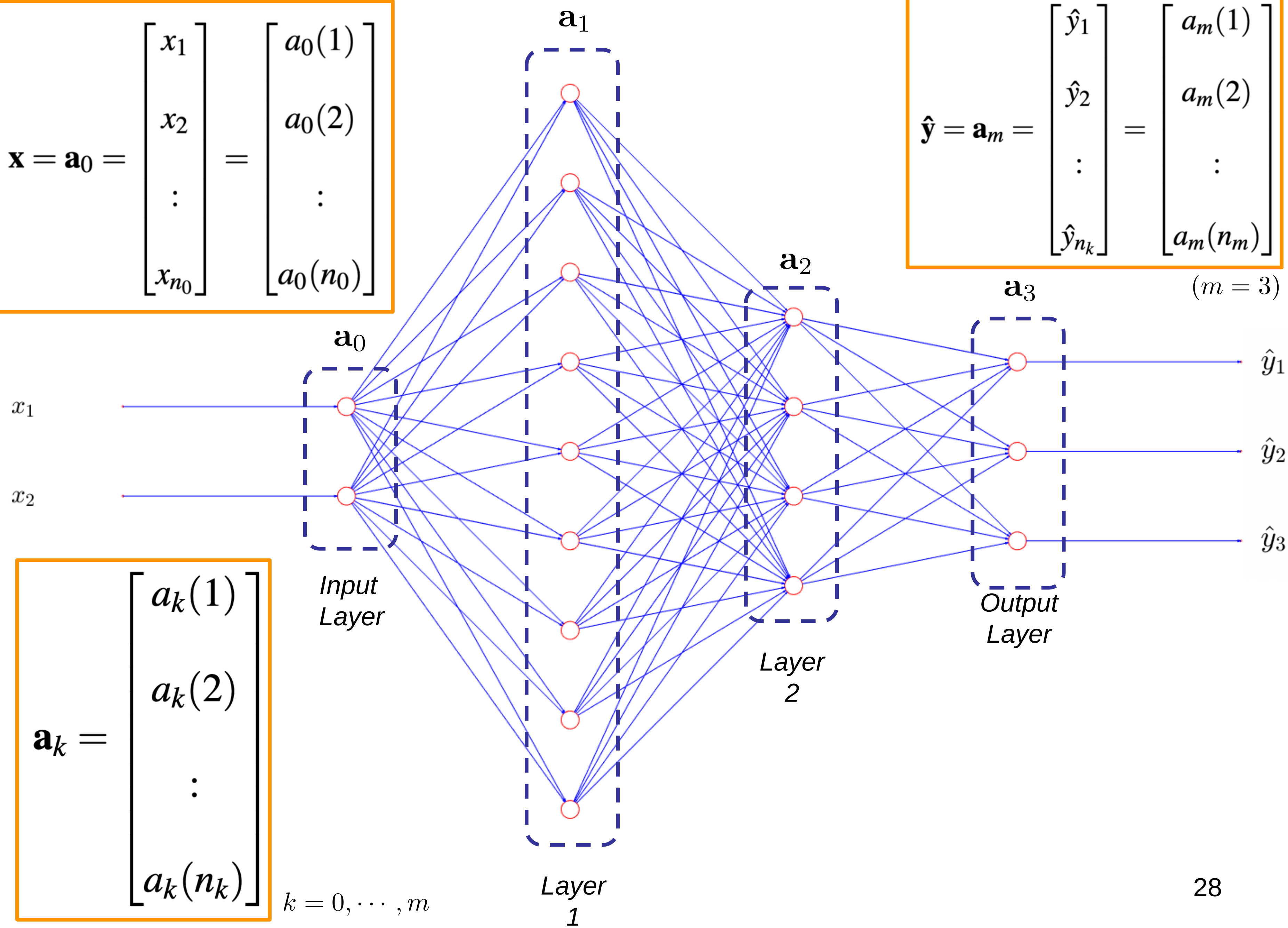
x_1
x_2
y_1
y_2
y_3


x_1
x_2
y_1
y_2
y_3
$f(\mathbf{x})$
$z = \mathbf{w}^\top \mathbf{x} + b$
$a = \sigma(z)$

$$\mathbf{x} = \mathbf{a}_0 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_0} \end{bmatrix} = \begin{bmatrix} a_0(1) \\ a_0(2) \\ \vdots \\ a_0(n_0) \end{bmatrix}$$

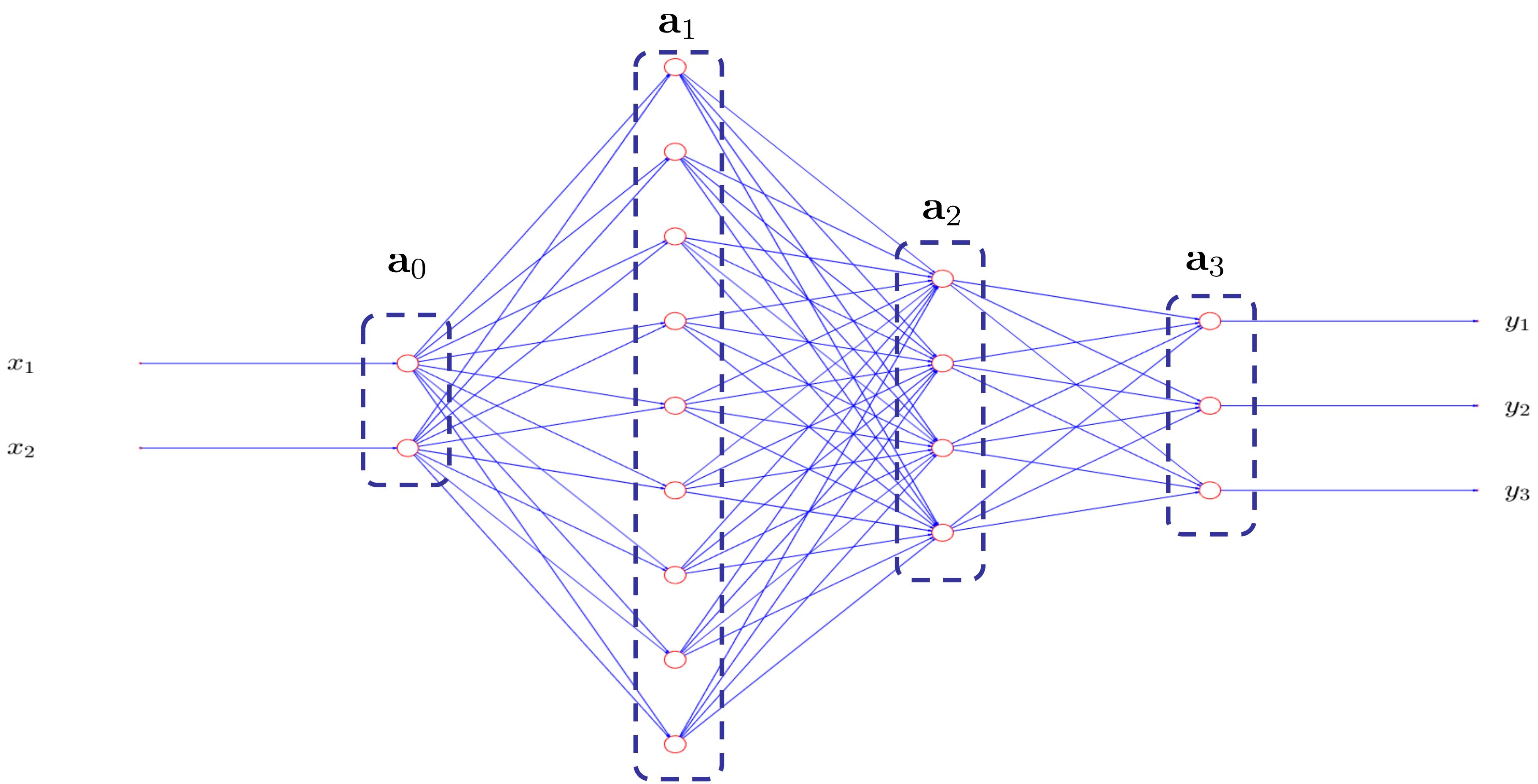
$$\hat{\mathbf{y}} = \mathbf{a}_m = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{n_k} \end{bmatrix} = \begin{bmatrix} a_m(1) \\ a_m(2) \\ \vdots \\ a_m(n_m) \end{bmatrix}$$

$(m = 3)$



$$\mathbf{a}_k = \begin{bmatrix} a_k(1) \\ a_k(2) \\ \vdots \\ a_k(n_k) \end{bmatrix}$$

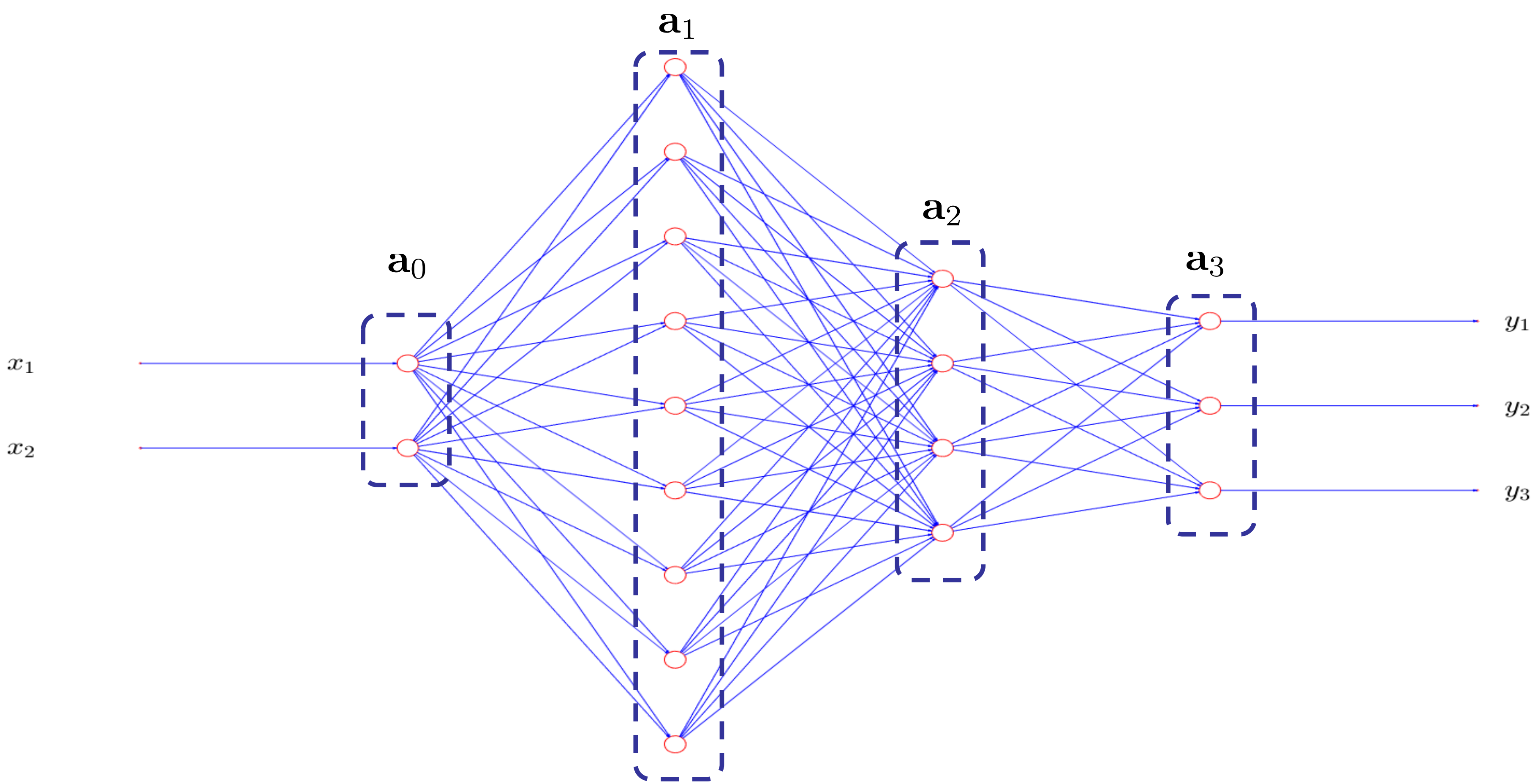
$k = 0, \cdots, m$



$$\mathbf{x} = \mathbf{a}_0 \rightarrow \mathbf{a}_1 \rightarrow \mathbf{a}_2 \rightarrow \hat{\mathbf{y}} = \mathbf{a}_3$$

$$\mathbf{z}_1 = \mathbf{W}_1\mathbf{a}_0 + \mathbf{b}_1 \qquad \mathbf{z}_2 = \mathbf{W}_2\mathbf{a}_1 + \mathbf{b}_2 \qquad \mathbf{z}_3 = \mathbf{W}_3\mathbf{a}_2 + \mathbf{b}_3$$

$$\mathbf{a}_1 = \sigma(\mathbf{z}_1) \qquad \mathbf{a}_2 = \sigma(\mathbf{z}_2) \qquad \mathbf{a}_3 = \sigma(\mathbf{z}_3)$$

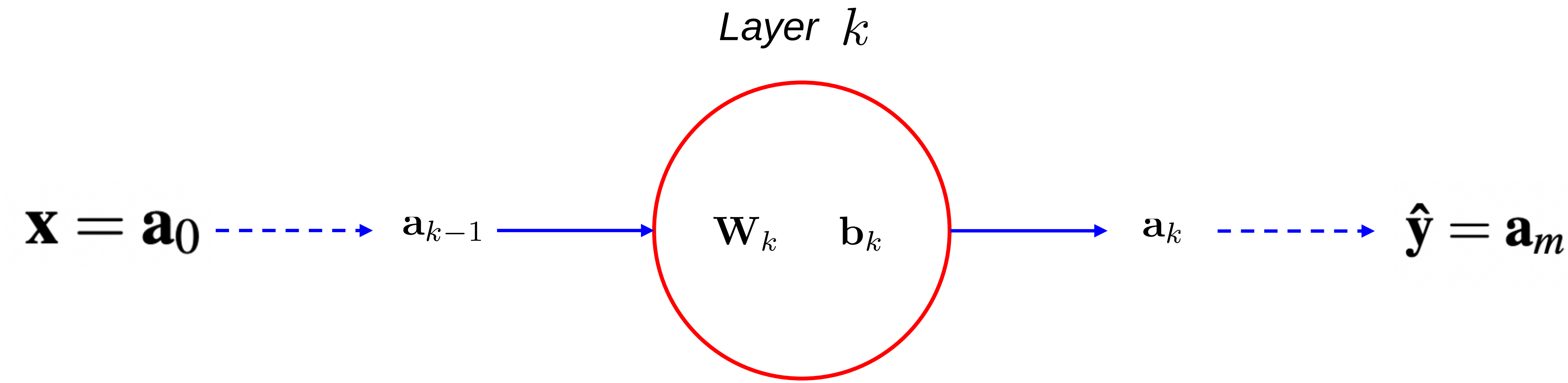


$$\mathbf{x} = \mathbf{a}_0 \rightarrow \mathbf{a}_1 \rightarrow \mathbf{a}_2 \rightarrow \hat{\mathbf{y}} = \mathbf{a}_3$$

$$\mathbf{z}_k = \mathbf{W}_k \mathbf{a}_{k-1} + \mathbf{b}_k$$

$$\mathbf{a}_k = \boldsymbol{\sigma}(\mathbf{z}_k) \qquad k = 1, \cdots, 3$$

Forward Propagation



$$\mathbf{z}_k = \mathbf{W}_k \mathbf{a}_{k-1} + \mathbf{b}_k$$

$$\mathbf{a}_k = \boldsymbol{\sigma}(\mathbf{z}_k)$$

$$k = 1, \cdots, m$$

Training

Es necesario estimar:
(los pesos y 'bias' de todas las capas)

$$\Theta = \{\theta_k\}_{k=1}^{m}$$

$$\theta_k = (\mathbf{W}_k, \mathbf{b}_k)$$

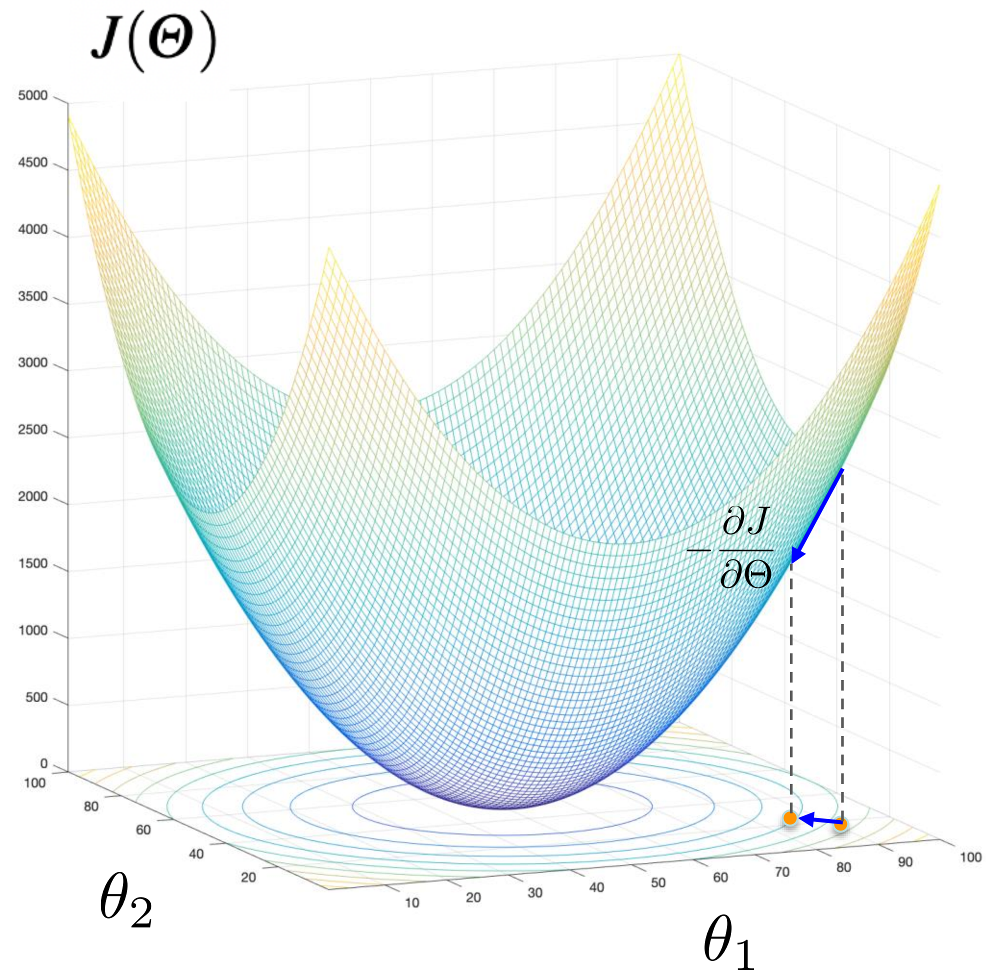
Se minimiza entonces la Función Objetivo:

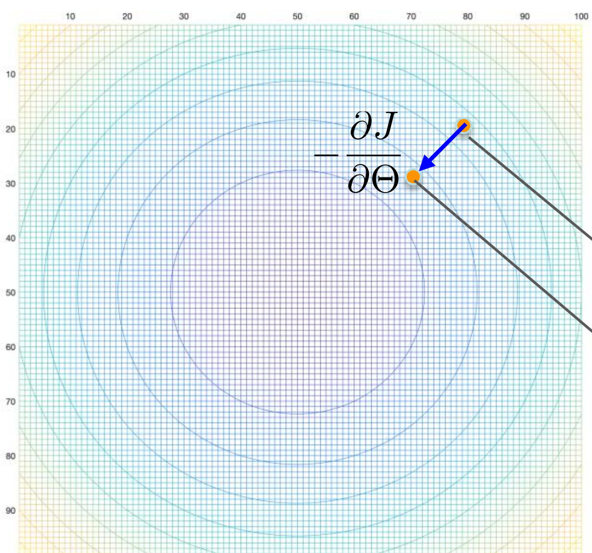
$$J(\Theta) = \frac{1}{N}\sum_{i=1}^{N} f_{\text{loss}}(\hat{\mathbf{y}}_i, \mathbf{y}_i) \rightarrow \textbf{min}$$

$$\frac{1}{2}||\hat{\mathbf{y}}_i - \mathbf{y}_i||^2$$

Ejemplo de función de pérdida

Minimización usando métodos de Gradiente

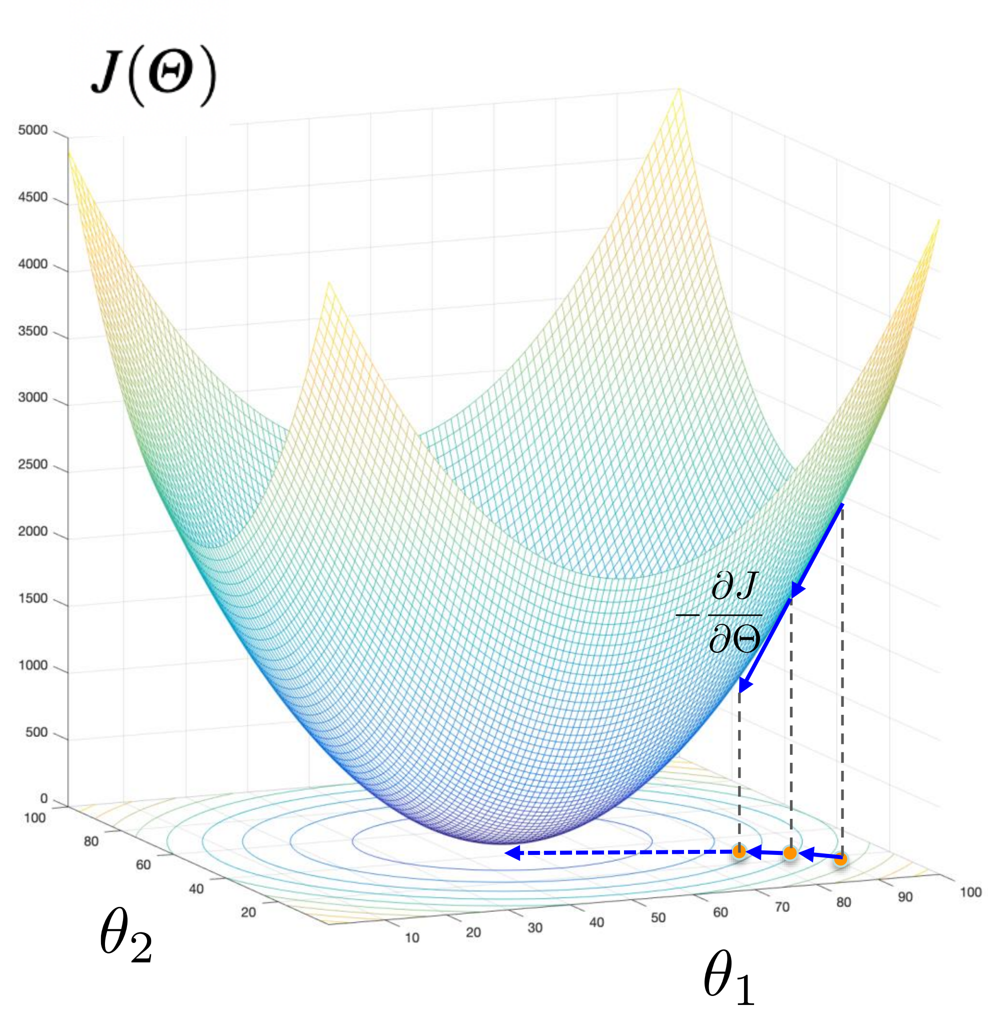


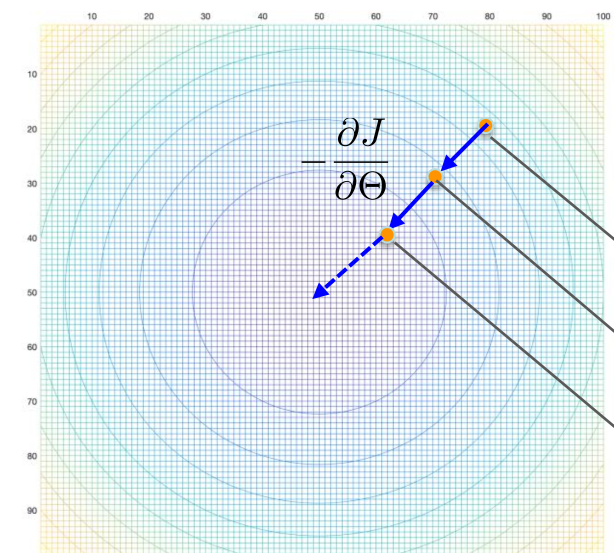


Valor inicial

Valor siguiente

Minimización usando métodos de Gradiente





Valor inicial

Valor siguiente

Valor siguiente

$$\Theta_r = (\theta_1, \theta_2)_r$$

$$\Theta_{r+1} = \Theta_r - \alpha \frac{\partial J}{\partial \Theta}$$

… iterar hasta converger

Algoritmo de Entrenamiento

1. Inicio de los parámetros con valores aleatorios:

$$\Theta = \{\theta_k\}_{k=1}^{m}$$

$$\theta_k = (\mathbf{W}_k, \mathbf{b}_k)$$

$$\mathbf{W}_k := \text{random matrix}(n_k \times n_{k-1})$$

$$\mathbf{b}_k := \text{random vector}(n_k \times 1)$$

Algoritmo de Entrenamiento

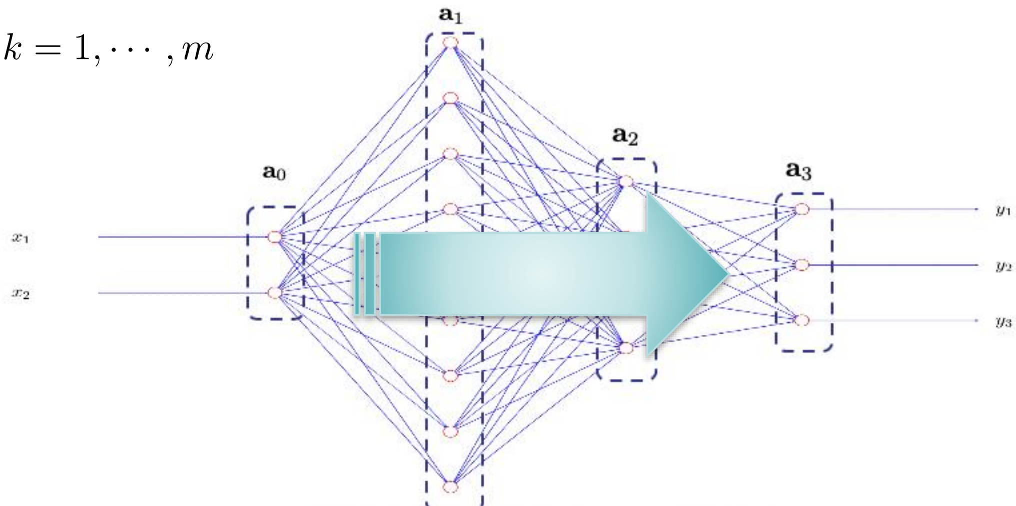
2. Forward Propagation para cada muestra:

$$\mathbf{z}_{k,i} = \mathbf{W}_k \mathbf{a}_{k-1,i} + \mathbf{b}_k$$

$$\mathbf{a}_{k,i} = \sigma(\mathbf{z}_{k,i}).$$

$i = 1, \cdots, n$

$k = 1, \cdots, m$



$\mathbf{x} = \mathbf{a}_0 \rightarrow \mathbf{a}_1 \rightarrow \mathbf{a}_2 \rightarrow \hat{\mathbf{y}} = \mathbf{a}_3$

Algoritmo de Entrenamiento

3. Cálculo de las derivadas parciales:

$$\Delta \mathbf{W}_k = \frac{\partial J}{\partial \mathbf{W}_k} \quad , \quad \Delta \mathbf{b}_k = \frac{\partial J}{\partial \mathbf{b}_k}$$

Algoritmo de Entrenamiento

4. Actualización de los parámetros usando un ‘learning rate’ α:

$$\mathbf{W}_k := \mathbf{W}_k - \alpha \Delta \mathbf{W}_k \quad , \quad \mathbf{b}_k := \mathbf{b}_k - \alpha \Delta \mathbf{b}_k$$

Algoritmo de Entrenamiento

5. Repetir desde el paso 2 hasta converger:

$$J(\mathbf{W}_1, \cdots, \mathbf{W}_m, \mathbf{b}_1, \cdots, \mathbf{b}_m) < \varepsilon$$

Eso es “todo” !

¿Cómo se calculan las derivadas parciales? del paso 3

$$\Delta \mathbf{W}_k = \frac{\partial J}{\partial \mathbf{W}_k} = \underbrace{\frac{\partial J}{\partial \mathbf{a}_k} \frac{\partial \mathbf{a}_k}{\partial \mathbf{z}_k}}_{\gamma_k} \underbrace{\frac{\partial \mathbf{z}_k}{\partial \mathbf{W}_k}}_{\mathbf{a}_{k-1}} = \gamma_k \, \mathbf{a}_{k-1}$$

$$\Delta \mathbf{b}_k = \frac{\partial J}{\partial \mathbf{b}_k} = \underbrace{\frac{\partial J}{\partial \mathbf{a}_k} \frac{\partial \mathbf{a}_k}{\partial \mathbf{z}_k}}_{\gamma_k} \underbrace{\frac{\partial \mathbf{z}_k}{\partial \mathbf{b}_k}}_{1} = \gamma_k$$

$$\gamma_k = \underbrace{\frac{\partial J}{\partial \mathbf{a}_k}}_{\text{input}} \underbrace{\frac{\partial \mathbf{a}_k}{\partial \mathbf{z}_k}}_{\sigma'_k} = \frac{\partial J}{\partial \mathbf{a}_k} \mathbf{a}_k (1 - \mathbf{a}_k)$$

Necesario para los cálculos

$$a = \sigma(z) = 1/(1 + e^{-z})$$

$$\sigma'(z) = a(1 - a)$$

¿Cómo se calculan las derivadas parciales? del paso 3

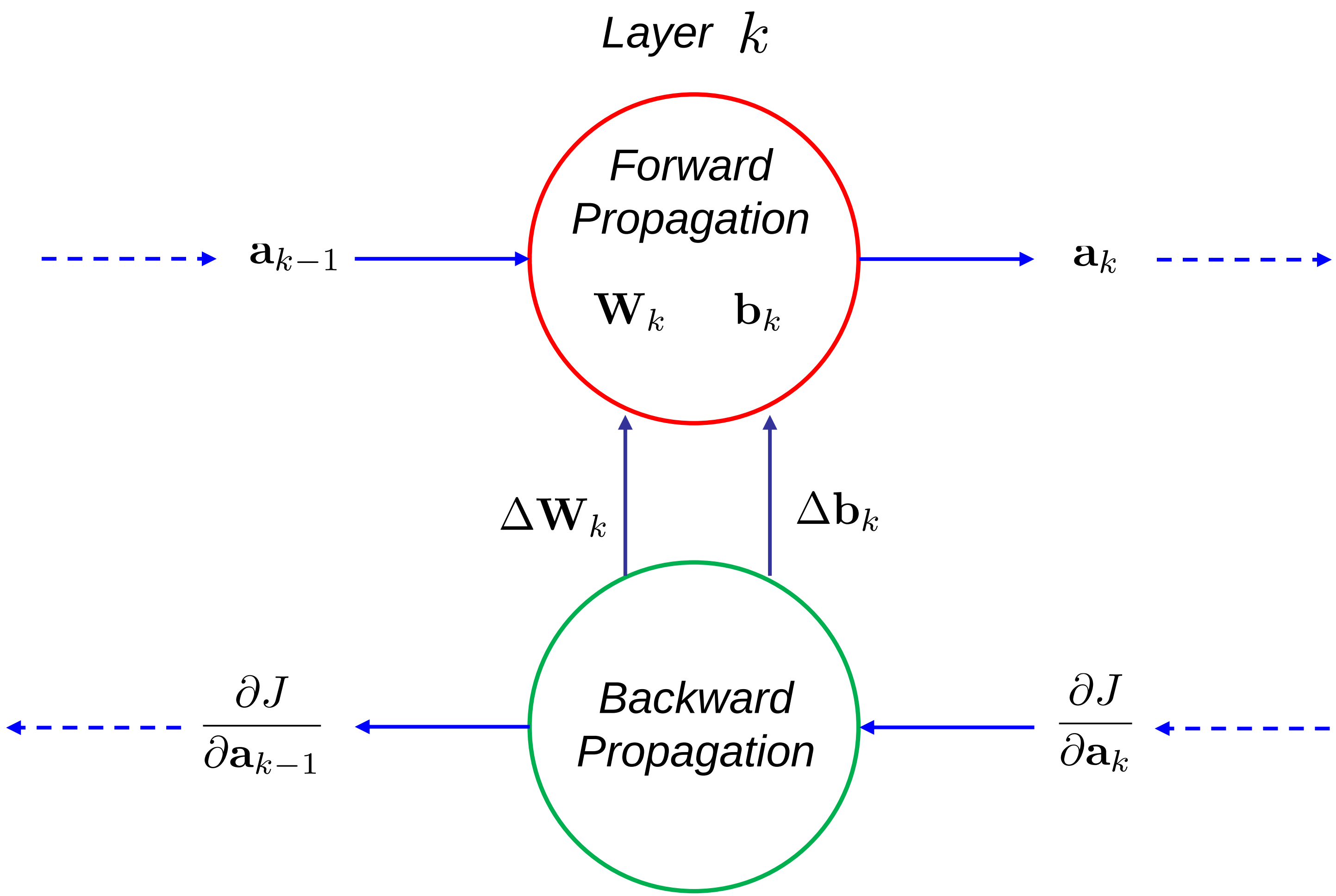
$$\frac{\partial J}{\partial \mathbf{a}_k}$$

Necesario para
los cálculos

$$\frac{\partial J}{\partial \mathbf{a}_m} = \frac{\partial}{\partial \mathbf{a}_m}\left\{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||\hat{\mathbf{y}}_i - \mathbf{y}_i||^2\right\} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{a}_{m,i} - \mathbf{y}_i)$$

$$\frac{\partial J}{\partial \mathbf{a}_{k-1}} = \underbrace{\frac{\partial J}{\partial \mathbf{a}_k}\frac{\partial \mathbf{a}_k}{\partial \mathbf{z}_k}}_{\gamma_k}\underbrace{\frac{\partial \mathbf{z}_k}{\partial \mathbf{a}_{k-1}}}_{\mathbf{W}_k} = \gamma_k\,\mathbf{W}_k$$

¿Cómo se calculan las derivadas parciales? del paso 3



Eso es “TODO” !!